Delphi Prism e ASP.NET
Reaproveite seus conhecimentos Win32 / Desktop para criar aplicações Web – Parte 4

Clube Delphi

Ano 9 :: Edição 121 :: R$ 11,90

DevMedia group

Boa Ideia
Formatação automática de código

Delphi 2010
Ainda não migrou?
Conheça boas razões para abandonar de vez o Delphi 7!

Persistência em Delphi Win32

Desenvolva aplicações de BD usando poderosos recursos de POO, como em .NET e Java

Brinde

Comprando esta edição ganhe 4 vídeo-aulas:

- MasterDataView: Relatório com 3 níveis de relacionamento com Rave Reports
- Delphi Prism: Gerenciamento de banco de dados com Visual Studio
- Delphi Prism: Facilitando a construção de strings de conexão
- Hash: Criando uma função MD5

jQuery
Integre suas aplicações com o Twitter e Twitpic

Quick Update
Herança com ActionManager

Delphi Prism
Acesso nativo ao Firebird

ISSN 1517990-7

Dificuldades em encontrar conteúdo sobre .Net para iniciantes?

A DevMedia simplifica pra você!

easy .net magazine

Easy .Net Magazine:

A primeira revista digital feita especialmente para quem está dando os primeiros passos em .Net!

Você encontrará exemplos e modelos fáceis e práticos, noções introdutórias e conceitos básicos.

Conheça as vantagens:

- Vídeos e apresentações "embutidos" nas páginas;
- Ilustrações, diagramas e modelos apresentados em PowerPoint;
- Entrevistas em vídeo;
- Códigos-fonte que podem ser copiados com CTRL+C/CTRL+V.

A Easy.Net não é feita em PDF ou Flash!
Ela utiliza um aplicativo 100% online.

DevMedia group

Acesse agora a edição Nº 1 totalmente grátis e comprove

www.devmedia.com.br/easynet

Easy.Net Magazine, seu curso inicial de .Net!

Cursos Online

Assinatura

.net magazine PLUS

Mais conteúdo .NET por muito menos!

A revista **.net Magazine** oferece para seus assinantes uma série de **Cursos Online** de alto padrão de qualidade .
Conheça abaixo os cursos já disponíveis.

Curso em destaque

Crie uma loja virtual completa

Confira neste curso Online como construir uma loja virtual completa no Visual studio 2005. Aprenda como criar o banco de dados, carrinho de compras, profile, página de erros e muito mais .

Confira o plano de aula completo:
www.devmedia.com.br/lojavirtual

A sua melhor opção de aprendizagem!
Assine a **.net Magazine** e comece já seu treinamento!
www.devmedia.com.br/assine

Outros cursos disponíveis: **www.devmedia.com.br/curso**

- Construindo relatórios com Crystal Reports e Visual Studio 2005
- Criando uma aplicação Web Completa
- Criando uma aplicação client/server no Visual Studio 2005
- Aprenda a criar um blog com ASP.NET
- Curso de C#
  * Curso em andamento

DevMedia group

Mais Informações: www.devmedia.com.br/central - Tel.: 21 3382-5038

Hospede seus sites de forma simples, rápida e ilimitada.

Assine agora mesmo e descubra por que Hospedix é a hospedagem de sites preferida entre os profissionais.

- Conheça também nossos planos profissionais com registro de domínio grátis e isenção de taxas anuais enquanto for cliente.

HOSPEDIX
HOSPEDAGEM PROFISSIONAL
www.hospedix.com.br

30 DIAS GRÁTIS

Ao assinar, digite o código de desconto exclusivo para leitores desta revista e ganhe 30 dias inteiramente grátis.

RVSTMEDIA

As promoções anunciadas não são cumulativas com outras promoções e são válidas apenas para clientes Hospedix ativos.

Sumário

Quick Update

06 – Herança com ActionManager
[*Daniel Sobrinho Laporte*]

Delphi

Novidades

08 – Delphi 2010
Novidades do IDE
[*Álison Bissoli Dias*]

Tutorial - Web

16 – Delphi Prism
Acesso nativo ao Firebird
[*Daniel Sobrinho Laporte*]

Boas Práticas

Boas Práticas

24 – Mapeamento Objeto / Relacional em Win32
Desenvolva aplicações de BD usando poderosos recursos de POO, como em .NET e Java
[*Paulo Quicoli*]

Easy Delphi

Easy Delphi - Mini-Curso – Tutorial – Web – Novidades

34 – Delphi Prism e ASP.NET
Reaproveite seus conhecimentos Win32 / Desktop para criar aplicações Web – Parte 4
[*Guinther Pauli*]

Easy Delphi - Boa Ideia

40 – Formatação de código-fonte inteligente
Formate qualquer código-fonte a sua maneira
[*Daniel Sobrinho Laporte*]

PHP

PHP/ Web/ Ajax

44 – Twitter, jQuery e PHP
Integrando seu Web Site com o Twitter e Twitpic
[*Jaison Schmidt*]

Brinde na web desta edição - 4 vídeos! Confira:

4 Vídeos

- **Hash: Criando uma função MD5**
  http://www.devmedia.com.br/post-17014-Hash--Criando-Funcao-MD5-Curso-Delphi-e-MySQL-Aplicacao-Financeira-Parte-27.html
- **MasterDataView: Relatório com 3 níveis de relacionamento com Rave Reports**
  http://www.devmedia.com.br/post-17215-MasterDataView--Relatorio-com-3-niveis-de-relacionamento-Curso-Rave-Reports-Parte-29.html
- **Delphi Prism: Gerenciamento de banco de dados com Visual Studio**
  http://www.devmedia.com.br/post-17242-Gerenciar-BD-com-Visual-Studio-Completo-Delphi-Prism-Desenvolvimento-Web-Parte-50.html
- **Delphi Prism: SqlConnectionStringBuilder, facilitando a construção de strings de conexão**
  http://www.devmedia.com.br/post-17389-SqlConnectionStringBuilder--Facilitando-a-construcao-de-strings-de-conexao-Curso-Completo-Delphi-Prism-Desenvolvimento-Web-Parte-56.html

Para acessar os vídeos utilize os seguintes dados
Login: **DVM.PL** *Senha:* **MOR4D**

Gostou das vídeo aulas? O portal *www.devmedia.com.br* possui mais de **2 mil vídeo aulas** e **dezenas de cursos online** sobre desenvolvimento de software! Agora você pode comprar as vídeo aulas que preferir e fazer sua própria combinação de vídeos! Saiba mais em *www.devmedia.com.br/creditos*

Dê seu feedback sobre esta edição!

A ClubeDelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre esta edição, artigo por artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Para votar, você vai precisar do código e senha de banca desta edição, que é:
Login: **DVM.PL** Senha: **MOR4D**

Clube Delphi

Ano 9 - 121ª Edição 2010 - ISSN 1517990-7 Impresso no Brasil

Corpo Editorial

Editor Geral
Guinther Pauli
guinther@devmedia.com.br

Editor Técnico
Paulo Quicoli
pauloquicoli@devmedia.com.br

Daniel Laporte
daniel.sobrinho@devmedia.com.br

Rodrigo Carreiro Mourão
rodrigocarreiro@rmfactory.com.br

Equipe Editorial
Vitor Luiz Rubio
vitorrubio@gmail.com

Rafael Stavarengo
faelsta@gmail.com

Capa e Diagramação
Romulo Araujo
romulo@devmedia.com.br

Coordenação Geral
Daniella Costa - daniella@devmedia.com.br

Revisão e Supervisão
Thiago Vincenzo - thiago.v.ciancio@devmedia.com.br

Distribuição
Fernando Chinaglia Dist. S/A
Rua Teodoro da Silva, 907
Grajaú - RJ - 206563-900

Atendimento ao Leitor

A DevMedia conta com um departamento exclusivo para o atendimento ao leitor. Se você tiver algum problema no recebimento do seu exemplar ou precisar de algum esclarecimento sobre assinaturas, exemplares anteriores, endereço de bancas de jornal, entre outros, entre em contato com:

Cristiany Queiroz – Atendimento ao Leitor
www.devmedia.com.br/mancad
(21) 3382-5038

Kaline Dolabella – Gerente de Marketing e Atendimento
kalined@terra.com.br
(21) 3382-5038

Publicidade

Para informações sobre veiculação de anúncio na revista ou no site e para fechar parcerias ou ações específicas de marketing com a DevMedia, entre em contato com:

Kaline Dolabella
publicidade@devmedia.com.br

Fale com o Editor!

É muito importante para a equipe saber o que você está achando da revista: que tipo de artigo você gostaria de ler, que artigo você mais gostou e qual artigo você menos gostou. Fique a vontade para entrar em contato com os editores e dar a sua sugestão!
Se você estiver interessado em publicar um artigo na revista ou no site ClubeDelphi, entre em contato com os editores, informando o título e mini-resumo do tema que você gostaria de publicar:

Guinther Pauli - Editor da Revista
guinther@devmedia.com.br

EDITORIAL

Nesta edição vamos derrubar um mito. Acho que essa é uma das maiores dúvidas dos desenvolvedores Delphi, principalmente daqueles que vêm do meio acadêmico ou que desenvolvem em outras tecnologias como Java e .NET, onde desenvolver de forma orientada a objetos é uma prática não só comum, mas obrigatória. Até hoje, tudo o que fazemos em Delphi se baseia em eventos. E programar exaustivamente com eventos torna a complexidade de um software muito grande. É muito fácil hoje no Delphi largar nosso famoso trio SqlDataSet, DataSetProvider e ClientDataSet e usando controles Data-Aware manipular dados de qualquer tabela do banco de dados. Isso é RAD. E provavelmente tudo o que você fará de código extra, que não é feito pelo dbExpress / VCL, escreverá em eventos. Até aí tudo bem.

Então vou aproveitar este editorial para relatar um email de um leitor da revista que teve exatamente a dúvida que todos têm e que vai ser esclarecida nesta edição, aqui está um trecho do email: "*Estamos na seguinte encruzilhada: criamos nosso modelo conceitual utilizando uma ferramenta de modelagem do Delphi, lá criamos todas nossas classes, ex.: cidades, clientes, pedidos etc. O mesmo gera a classe base (tipo o POJO do Java, POCO do .NET). Aí começam nossos problemas. Primeiro, existe algum framework tipo ORM que utilize tais classes para efetuar a persistência em banco de dados e a criação do mesmo? Segundo como utilizar tais classes juntamente do nosso tão adorado ClientDataSet e os componentes Data-Aware?*"

Eu também tive essa dúvida, ainda nos tempos acadêmicos. Resumindo, pediram para eu fazer meu diagrama de classes (modelo conceitual), imagine, por exemplo, o digrama abaixo (lado esquerdo). Depois, pediram para criar o famoso MER ou ainda DER (Diagrama Entidade Relacionamento), no lado direito. A grande questão é, o que eu faço com o da esquerda? Existe um grande "buraco" entre esses dois mundos! Eu não imaginava, mas a minha dúvida é a mesma que até hoje persiste entre muitos programadores Delphi: eu precisava de um framework de mapeamento objeto / relacional. Tudo bem, em .NET e Java isso é uma prática comum, mas e em Delphi Win32? Sim, é possível. E o ClientDataSet? Bom, no caso do framework que vamos apresentar, poderemos usar TDataSets, o que nos dá um certo potencial RAD por poder continuar usando controles Data-Aware. O melhor dos dois mundos.

Diagrama de Classes UML vs Modelo Entidade / Relacionamento

Ainda nesta edição, bons motivos para você tomar coragem e colocar sua caixa do Delphi 7 com CD de instalação no cofre e começar a usar o Delphi 2010. No nosso mini-curso de Delphi Prism, vamos aprender a criar consultas parametrizadas sem uma linha de código, e comprovar que desenvolver com ASP.NET é tão natural quanto usar a VCL (você não precisa aprender Java para desenvolver aplicações robustas para Web). E tem mais Delphi Prism! Veja como acessar de forma nativa o banco Firebird. Na sessão Easy, veja ainda como usar alguns utilitários para formatar automaticamente seu código-fonte Delphi. E finalizando, veja como integrar PHP, jQuery com o Twitpic e Twitter, o maior micro-blog do planeta. Boa leitura!

Guinther Pauli

guinther@devmedia.com.br

Certificado Delphi 3,5,6,7,2005,2006,Web,Kylix

Microsoft Certified: MCP,MCAD,MCSD.NET,MCTS,MCPD (ASP.NET,C#)

guintherpauli.blogspot.com - twitter.com/guintherpauli

ClubeDelphi PLUS

DevMedia group

A revista ClubeDelphi é parte integrante da assinatura ClubeDelphi PLUS.
Para mais informações sobre o pacote PLUS, acesse:
http://www.devmedia.com.br/clubedelphi/portal.asp

Chegou o Sistema de Créditos DevMedia

Agora o **conteúdo completo** do nosso site está **ao seu alcance!**

A partir de agora todas as **4000 vídeo-aulas** do site DevMedia podem ser compradas individualmente.

Economia - Você não precisa mais assinar oito produtos diferentes para ter acesso ao conteúdo completo do site!

Todas as vídeos que você sempre quis ver a partir **R$ 0,99!**

Saiba mais sobre o Sistema de Créditos!

DevMedia GROUP

www.devmedia.com.br/creditos

Quick Update:
Herança com ActionManager

Daniel Sobrinho Laporte
daniel.sobrinho@devmedia.com.br
daniel.laporte@gmail.com
www.twitter.com/daniel_laporte
Colunista e editor técnico da Revista ClubeDelphi. Graduado em Sistemas de informação e pós-graduado em Administração de Bancos de Dados pela Universidade de Franca. É analista de sistemas e desenvolvedor Delphi desde 2005 desenvolvendo aplicações cliente/servidor e multicamadas.

Utilizar o ActionList para centralizar as ações do formulário é uma eventual prática dos desenvolvedores. Unido ao conceito de herança visual, este componente pode tornar seu desenvolvimento muito mais produtivo, pois centraliza os eventos em suas ações, evitando a duplicidade de código e auxiliando no reuso de todo o código já existente. E é claro que com a chegada das novas versões do Delphi nós queremos deixar nossos projetos compatíveis com o que há de mais moderno, como os novos recursos de suporte a Touch e Gestures por exemplo. Para isso, a Embarcadero recomenda desde a versão do Delphi 2009 que os projetos utilizem o ActionManager no lugar do ActionList, porém o grande problema se dá quando tentamos criar uma herança visual utilizando este componente. Veja o erro obtido na **Figura 1**.

Tradicionalmente utilizamos um ActionList para centralização dos métodos, principalmente em cadastros, como mostra a **Figura 2**.

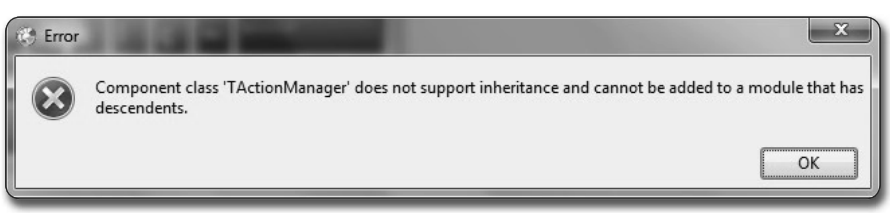


Figura 1. Problema ao trabalhar com herança visual e ActionManager

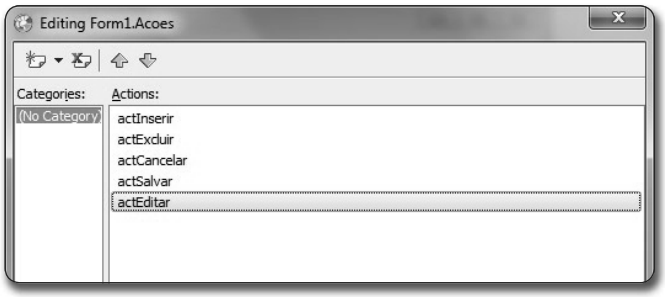


Figura 2. Exemplo de uso do ActionList

Se você já possui conhecimento sobre os conceitos de encapsulamento, conseguirá utilizar as rotinas de uma forma muito simples, ainda assim, não será possível utilizar a herança do ActionManager, porém você poderá fazer algo parecido com o que vimos na **Figura 2**. O que se sugere aqui é que possamos criar estes métodos em nossa classe base (formulário de herança), criando-os no escopo protected, como ilustrado no exemplo da **Listagem 1.**

Listagem 1. Criando os métodos no formulário de herança

```
unit untHeranca;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ActnList;

type
  TfrmHeranca = class(TForm)
  private
    { Private declarations }
  protected
    procedure Salvar;
    procedure Excluir;
    procedure Editar;
    procedure Cancelar;
    procedure Inserir;
  public
    { Public declarations }
  end;
```

Note no código da **Listagem 1** que criamos as procedures referentes aos nossos métodos. Desta forma criaremos um novo formulário baseado no formulário de herança. O escopo protected, para quem não conhece, permite que métodos, propriedades ou campos de uma classe pai posicionados nela sejam visíveis apenas às classes que herdam dessa classe pai, diferente do public onde todos os métodos são visíveis pela aplicação. Criando um formulário descendente do código da **Listagem 1** poderemos fazer algo como mostrado na **Figura 3** e na **Listagem 2.**

Assim poderemos utilizá-los dentro de um ActionManager, fazendo uso de boas práticas, reaproveitando seu código e é claro, permitindo o uso do que há de mais novo no Delphi 2010. Grande abraço e até a próxima!

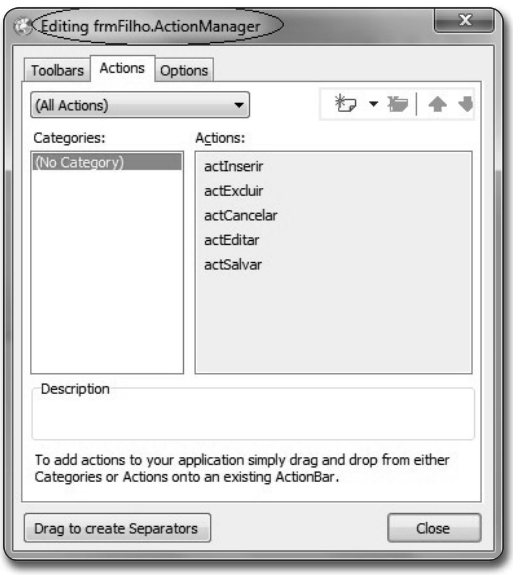


Figura 3. Utilizando métodos no ActionManager

Listagem 2. Codificando os métodos protegidos no ActionManager

```
procedure TfrmFilho.actCancelarExecute(Sender: TObject);
begin
  inherited;
  Cancelar;
end;

procedure TfrmFilho.actEditarExecute(Sender: TObject);
begin
  inherited;
  Editar;
end;

procedure TfrmFilho.actExcluirExecute(Sender: TObject);
begin
  inherited;
  Excluir;
end;

procedure TfrmFilho.actInserirExecute(Sender: TObject);
begin
  inherited;
  Inserir;
end;

procedure TfrmFilho.actSalvarExecute(Sender: TObject);
begin
  inherited;
  Salvar;
end;
```

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Delphi 2010

Novidades do IDE

O Delphi em toda sua história sempre evoluiu para suprir rapidamente as exigências das novas arquiteturas do mercado. Em seu lançamento, o Delphi 1 já representou uma revolução para o desenvolvimento de softwares, evoluiu da linguagem Pascal, sendo uma linguagem orientada a objetos e ao mesmo tempo a eventos. O Delphi já naquela época suportava o desenvolvimento Client/Server, possuía compilador nativo e extremamente rápido (16 bits) e introduziu o conceito de ambiente de programação RAD (Rapid Application Development) orientada a objetos. Isto em 1995.

Em 1996, após o lançamento do Windows 95, surge a versão 2, já em 32 bits, porém mantendo a compatibilidade com a versão 16 bits. Uma grande preocupação, que os desenvolvedores de todas as versões sempre tiveram, foi manter a máxima compatibilidade de uma versão para outra, o que facilitou muito a evolução e migração de projetos e componentes. Melhorias foram disponibilizadas como o tipo Variant de dados, e a herança de formulários, recurso que facilitou o desenvolvimento de aplicações Desktop eliminando o tempo que o desenvolvedor ficaria duplicando telas e códigos similares, como as de cadastros que somente fizeram referências a um formulário e códigos comuns. Nesta versão também surgiram os Data Modules, que como tudo mundo sabe, é essencial em qualquer tipo de projeto.

A Internet começou a ganhar maior relevância em 1997, e o foco dos desenvolvedores de todo o mundo foi o desenvolvimento para rede, e a resposta foi o Delphi 3, com suporte a desenvolvimento Web utilizando CGI e ISAPI/NSAPI. Uma das tecnologias para desenvolvimento de aplicações em rede foi o N-Tier (multicamadas) DCOM com "Midas". Este foi considerado um milagre no que se diz em relação ao desenvolvimento multicamadas. Esta versão ainda trouxe o Code Insight, permitiu a depuração de DLLs, disponibilizou novos componentes como o Decision Cube e o TeeChart.

Em 1998, com o Windows 98, era esperado que fosse lançada uma nova versão, o Delphi 4 surgiu e adequou a linguagem à API da nova versão do Windows, e ainda ofereceu suporte ao padrão CORBA e ao Oracle 8. Apareceram assistentes para construção de controles Active X e inovou o Debug, permitindo a depuração remota de aplicações. Foi disponibilizado o AppBrowser e um grande destaque na linguagem foi o suporte a overloading de métodos.

Em 1999 a onda foi a utilização da tecnologia XML (eXtensible Markup Language), como resposta, o Delphi 5 foi lançado promovendo grande facilidade para os desenvolvedores trabalharem com esse novo padrão de arquivos. A Borland, detentora da ferramenta Delphi até então, sabia que devia englobar todas as tecnologias que estavam em alta no que dizia a desenvolvimento, não se retraindo à própria tecnologia e sim agregando as tendências para suprir necessidades, e assim vieram componentes para a tecnologia de acesso a dados ADO. Nessa versão ainda foram lançados o Team Source e Translation Suite.

Em 2001, surge o Delphi 6, com melhorias no IDE, XML, compilador, COM/aclives, e suporte a bancos de dados. Linux era

Resumo DevMan

De que trata o artigo:

Este artigo trata das novas facilidades e recursos disponibilizados pela Embarcadero na última versão do Delphi. Veremos um pouco sobre as novidades de linguagem, recursos do IDE, novos componentes e suas perspectivas.

Para que serve:

Este artigo serve para o programador conhecer bem seu ambiente de desenvolvimento para que saiba tirar proveito ao máximo de todos os recursos, proporcionando assim um ganho de produtividade imenso.

Em que situação o tema é útil:

Este artigo é útil como um excelente guia para aqueles que já dominam ou utilizam o Delphi em suas versões anteriores e precisam de uma forma rápida e prática aprender os novos recursos existentes tanto no IDE do Delphi 2010 quanto na linguagem e frameworks.

Delphi 2010:

Conheceremos neste artigo os novos recursos do Delphi 2010, fazendo com que você tenha uma boa impressão da versão atual do produto e assim pense no que ainda está por vir. Conheceremos o poder de um IDE RAD que possui o que há de melhor em um apanhado geral de versões. Veremos por exemplo como localizar códigos em vários arquivos, novos componentes e tecnologias, particularidades da compilação e do debug, novos recursos para debug de Threads, saberemos o que mudou no visual do IDE e como permitir que ele tenha a aparência da versão mais adorada do Delphi de todos os tempos, a versão 7.

o assunto do momento, e o Delphi 6 foi lançado com o conceito inovador de "Cross-platform" nativa, de forma que o mesmo código fonte do Delphi podia ser compilado no Delphi para Windows e no Kylix para Linux. Nasceu nesta versão o Kylix, que foi o Delphi para Linux, o DataSnap e WebSnap.

Uma tecnologia que trouxe uma grande transformação no Delphi foi o .NET, em 2002 a Borland lançou o Delphi 7 Studio, com o que David Intersimone chamou de "Delphi for .Net Preview Edition", em que o pacote Delphi possuía um CD separado com um compilador e ferramentas em linha de comando, documentação e bibliotecas VCL implementadas sobre .NET. Já se podia portar os projetos Delphi com algumas mudanças para atender convenções de nomeação de algumas units, recompilar e ter uma aplicação .NET. Esse preview foi lançado e prometido para o ano seguinte a versão completa .NET. Foi incorporado o IntraWeb, o DataSnap foi liberado nas versões Enterprise e Architect com licenças ilimitadas do Midas, possibilitando a distribuição de aplicações DataSnap/Midas a custo zero, porém neste ponto o DataSnap perdeu o suporte a conexões CORBA. O Modelmaker veio nas versões Architect e Enterprise com suporte a Refactoring e Design Patterns.

Dando continuidade à evolução para .NET iniciada na versão 7, a versão 8 foi lançada em 2003, com o nome de "Delphi 8 for the Microsoft .NET Framework" (resultado do projeto de codinome Octane), deu a oportunidade de trabalhar com dois grandes frameworks: VCL.NET que incluiu a maioria das classes VCL adaptadas para trabalhar com .NET e o Framework FCL com suporte ao desenvolvimento de aplicações ASP.NET (Web Forms e Web Services) e aplicações Windows Forms.

O compilador do Delphi 8 gerava somente código gerenciado deixando as aplicações mais seguras e menos sujeitas a falhas, para que isso fosse possível foram descontinuados alguns tipos e recursos do Delphi como o PChar, Pointer e Real48, conversão de ponteiros para inteiros, diretivas absolute e asm. Integraram-se à versão 8 o CaliberRM, Together, Optimizeit e StarTeam.

Em 2005 foi lançado o Delphi 2005, de codinome Diamondback, com o IDE suportando linguagens diferentes, permitindo a construção de aplicações .NET, Win32 e C#, sendo os 3 ambientes integrados ao mesmo IDE. Permitiu o desenvolvimento Web (disponibilizando ASP.NET com C# e WebBroker).

Ainda no final de 2005, surgiu o "Borland Developer Studio 2006", do projeto conhecido pelo codinome "Dexter", comportando 4 produtos em um só, ao adquirir qualquer um dos quatro produtos, ganhava o direito de utilizar os outros. Esses produtos eram "Delphi 2006 – Win32", "Delphi 2006 - .NET", "C# Builder 2006" e "C++ Builder 2006", foi a décima versão do Delphi. Esta versão deixou para trás o suporte à CLX, não incluía suporte ao .NET Compact Framework e ainda não suportava o .NET Framework 2.0. Depois de muitas críticas ao Delphi 2005 esta versão ganhou algumas melhorias para sanar esta parte, uma delas um novo Memory Manager criado por Pierre Le Riche, alocando e gerenciando memória mais rapidamente, com a aplicação desta tecnologia o próprio IDE que foi construído em Delphi ficou mais rápido.

O Delphi 2007 Win32, primeiro lançamento do Delphi em poder da CodeGear, surgiu como um "colírio" para os desenvolvedores de sistemas Desktop, incluiu suporte ao Windows Vista. A VCL para Web permitiu a aplicação da tecnologia AJAX, e passou a suportar o SSL (Secure Socket Layer), protocolo essencial para comunicação com Web Services de nota fiscal eletrônica brasileira. Surgiu o DBX4. No IDE, foi possível incluir macros e comandos DOS para serem executados antes ou depois de compilar o projeto, dar nomes a configurações de Builds, e em comparação com as versões anteriores, a versão estava incrivelmente ágil e leve.

A próxima versão após o Windows 2007 foi a 2009, o RAD Studio 2009 se manteve leve, trouxe o Class Explorer que possibilita a visualização de hierarquia de classes, interfaces dos projetos, adicionar propriedades, métodos e variáveis. O suporte a UNICODE foi um grande avanço. Generics que antes era um recurso apenas do .NET foi disponibilizado para Win32. Foi incluída também a tecnologia de métodos anônimos, partes de códigos que podem ser associadas a variáveis e serem utilizadas como parâmetros em métodos. O novo DataSnap se desligou da dependência à tecnologia COM, embora continuasse compatível. Outro marco desta versão foram as melhorias na integração com AJAX e o suporte a SilverLight.

No Delphi 2010, houve mais de 120 melhorias e novos recursos no IDE, Debugger, Open Tools API (OTA), compilador, lin-

guagem, VCL, acesso a bancos de dados e DataSnap. Após acompanharmos toda esta linha do tempo que descreveu toda história e evolução Delphi, iremos conhecer as novidades desta nova versão, que tem tudo para se tornar a melhor versão de todos os tempos, abordando recursos novos de linguagem, IDE, novidades, dicas e muito mais.

File Reopen

Além dos projetos que trabalhamos, estamos sempre abrindo exemplos, fontes de componentes, arquivos de outros projetos para consultas, e com isso, nos perdemos em meio a tantos diretórios. Para ajudar a organizar o acesso aos últimos projetos e arquivos abertos, foi adicionada ao IDE a opção File>Reopen>Properties, conforme mostra a **Figura 1**.

Esta tela permite que configuremos o número máximo de projetos a serem listados na opção File>Reopen, assim como o número máximo de arquivos. Na janela de propriedades, podemos limpar toda a lista de históricos clicando em "Clear" ou apagar somente um de cada vez selecionando a opção "Delete". O recurso "Delete Nonexistent Files" retira da lista arquivos não existentes, realizando uma checagem se aquele diretório é válido e removendo, por exemplo, arquivos que foram abertos de um disco removível.

Outra forma de organizar o acesso aos projetos mais utilizados é trabalhar com a janela Welcome Page, que é encontrada no menu View>Welcome Page. Nesta página, encontramos em "Recent Projects" os mesmos projetos de File, Reopen.

É possível marcar projetos como favoritos, e criar grupos para ajudar a organizá-los. Para criar um grupo basta acessar o no menu da Welcome Page>Projects>Manage Favorites e em "Create New Favorite Group" informando o nome do grupo a ser criado na opção "Group Title" e clicando em "Create" em seguida. Para adicionar qualquer projeto a um grupo de favoritos, basta acessar "Make me a Favorite", e em "Manage Favorites" mover o projeto para o grupo criado, e selecione "move", escolhendo o grupo na caixa "to". Em "Manage Favorites" é permitido tanto gerenciar os grupos como retirar projetos de favoritos.

O menu News é a porta de entrada de notícias, na opção "resources" temos links importantes como para o EDN (Embarcadero Developer Network) e portais de parceiros da Embarcadero, como a Glyfix (veja sessão Links) que disponibiliza bibliotecas de ícones, muitos deles gratuitos. Em "Documentation", temos acesso à documentação do sistema, assim como à sua base de dados de ajuda.

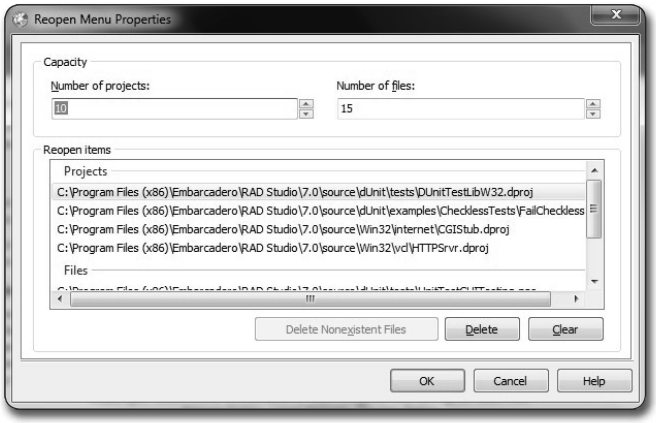


Figura 1. Janela Reopen Menu Properties

Através da Welcome Page podemos realizar toda a manutenção de nossos projetos, organizando-os, bem como acessar de forma rápida os últimos projetos.

IDE Insight

É muito comum perdermos tempo procurando por uma opção em qualquer sistema, aquela opção que fica dentro de uma aba de uma janela que só abre se for configurada outra janela e por aí vai. Com o IDE Insight você consegue localizar e acessar qualquer informação de qualquer janela ou menu do Delphi, inclusive localizar componentes.

A janela IDE Insight (**Figura 2**) pode ser acessada em tempo de projeto pelo atalho F6 ou pelo menu Search>IDE Insight.

Figura 2. Janela IDE Insight

No botão lupa podemos marcar/desmarcar onde serão procurados os termos de sua busca textual, conforme ilustra a **Tabela 1**.

Grupos de recursos do Delphi para busca	
Preferences	Opções do Delphi.
Project Options	Opções do projeto.
Components	Componentes.
Commands	Comandos dos botões e menus.
New Itens	Itens constantes na tabela New Itens, como Form e Unit.
Open Files	Arquivos atualmente abertos no IDE.
Recent Files	Arquivos abertos recentemente.
Recent Projects	Projetos acessados recentemente no IDE.
Component Palette	Localiza componentes por parte do nome da paleta.
Code Templates	Templates de códigos.
Desktop SpeedSettings	Configurações rápidas da área de trabalho.
Projects	Projetos abertos.
Files	Arquivos do projeto.
Forms	Formulários de projetos.

Tabela 1. Termos de busca do IDE Insight

Maior suporte à imagens

Agora a VCL suporta mais tipos de imagens como GIF image, Portable Network Graphics (png), JPEG Image file (jpg, jpeg), Bitmaps (bmp), TIFF images (TIF, TIFF), ícones (ico), metafiles (emf, wmf) etc. Veja um exemplo do componente TImage aberto na **Figura 3.**

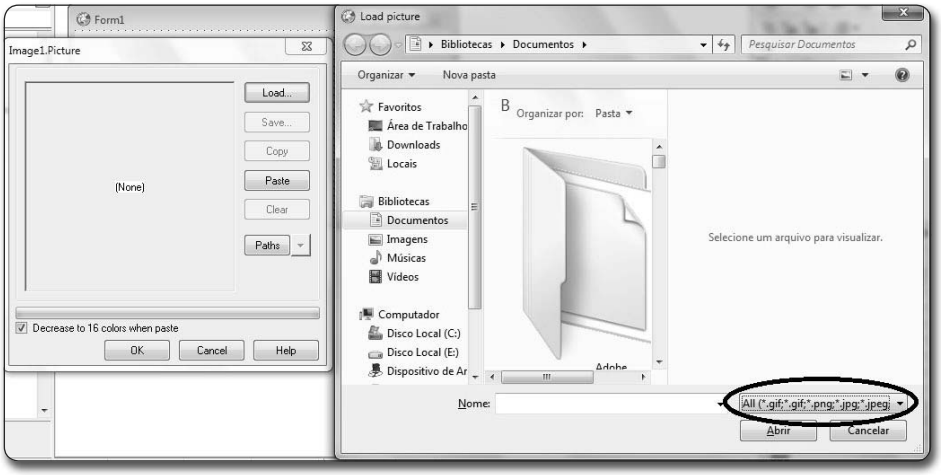


Figura 3. Suporte a novos formatos de imagem

Delphi 2010 com layout do Delphi 7

Com o visual padrão do Delphi 2010, que teve início na versão 2005, todas as janelas estão integradas umas às outras, por exemplo, a paleta de componentes está do lado direito incorporada ao IDE, este padrão é chamado de padrão "Dockável", pois permite realizar o Dock das janelas no próprio IDE, muito diferente de como era no Delphi , que possuíamos janelas "flutuantes". Pensando no padrão do Delphi 7, que foi a versão mais bem sucedida do IDE, a Embarcadero disponibilizou recursos de customização que permitem a personalização total do ambiente, possibilitando assim que você consiga ter o IDE conforme você desejar, unindo o que há de melhor nas versões. Desta forma diminui também a curva de adaptação dos usuários de Delphi 7 que desejam adotar a nova versão.

Para começar, iremos desvincular a janela Form Designer do Code Editor, para isso vá em Tools>Options>Environment Options>VCL Designer e desmarque a opção "Embedded Designer", ou utilize a tecla de atalho F6 do IDE Insight e ele automaticamente localizará a opção que estamos buscando. Para aplicar esta alteração, devemos reiniciar o Delphi. Após reiniciar crie um novo projeto e veja que o Form Designer não está mais embutido no Code Editor, sendo agora duas janelas flutuantes.

Agora vamos fazer com que toda a aplicação fique flutuante como no Delphi 7, para isso em Layout escolha a opção "Classic Undocked" e veja que todos os itens da IDE estão em janelas flutuantes. Caso a paleta de componentes não esteja disponível, podemos exibi-la clicando com o botão direito na barra de menus e escolhendo a opção "Components", assim a lista de componentes será exibida e já integrada à janela principal, como no Delphi 7. Posicione a barra de componentes na parte inferior da janela. Note que há um filtro para facilitar a busca pelos componentes.

Se de algum modo, a grande quantidade de janelas flutuando em seu IDE lhe causou uma certa confusão, podemos fechar as janelas que não temos costume de utilizar ou agrupá-las umas às outras, arrastando-as umas sobre as outras. É possível fazer isso com as janelas Structure, Object Inspector, Project Manager ou com o Debug, por exemplo, proporcionando total liberdade para organização de janelas flutuantes do IDE.

Como toque final, clique com o botão direito sobre qualquer propriedade no Object Inspector e selecione a opção Arranje, que permitirá alterar entre "By Category", que lhe permitirá agrupar as janelas ou "By Name", que apenas ordenará conforme o nome. Vale lembrarmos que esta já é uma opção antiga do Delphi.

Compilação em segundo plano

Dependendo do tamanho do software, os componentes utilizados no mesmo ou o desempenho da máquina, perdemos muito tempo compilando nossos projetos, a maioria dos IDEs disponíveis para diversas linguagens simplesmente travam durante o processo de compilação, ou somente exibem uma janela com o status do processo. Nesta versão do IDE Delphi é permitido realizar compilações de projeto em segundo plano, permitindo que façamos outras tarefas enquanto isso.

Nota do DevMan

O efeito Dock é caracterizado por permitir a customização do ambiente, selecionando a janela, arrastando-a para um local específico e fazendo com que ela se una ao local escolhido. Este efeito é muito comum nos IDEs atuais como Delphi e Visual Studio e torna o ambiente altamente customizável.

Ao efetuar uma compilação, o Delphi automaticamente cria uma nova Thread para a realização desta tarefa, assim o IDE continua respondendo aos seus comandos, permitindo que o desenvolvedor continue aceitando alterações no código do projeto normalmente no tempo em que o projeto está sendo compilado.

Suponhamos que o compilador leia um código de uma linha que você está alterando, e que ainda não acabou de alterar, isto causaria um erro, certo? Errado! Ao iniciar a compilação do projeto o IDE faz uma cópia em memória de todos os códigos e formulários de todos os projetos abertos, logo, as alterações durante a compilação não surtirão efeito, o compilador montará o arquivo binário baseado somente nos arquivos que estavam no momento em que se iniciou o processo de compilação.

Por ser um recurso novo e revolucionário, por padrão está desabilitado ao instalar o Delphi, para ativá-lo devemos teclar F6 e no IDE Insight buscar por "Background Compilation" e dar um duplo clique sobre o recurso encontrado. Será exibida a tela disponível em Options>Environment Options, marque a opção "Background

Compilation" que indica ao IDE que ele deve trabalhar com compilações em segundo plano. Outra opção que temos relacionada na mesma tela é de escolher a prioridade da Thread de compilação, na opção "Priority", mas cuidado, ao escolher uma prioridade High poderá estar suprimindo o processo utilizado para possibilitar a edição direcionando a utilização do processador mais para a compilação. Com a opção "Background Compilation" a tela Build que exibe o status da compilação perde sua opacidade, ficando transparente o bastante para não poluir visualmente o ambiente de edição, possibilitando que o IDE continue disponível para desenvolvimento mesmo durante a exibição do status de compilação.

Depuração e congelamento de Threads

Um novo recurso é o suporte ao Debug de Threads, é possível nesta versão realizar o acompanhamento da execução da Thread, o congelamento de uma Thread ou de todas as outras. A janela "Thread status" pode ser acessada no menu View>Debug>Threads. Durante o Debug, para congelar um Thread nesta janela, clique com o botão direito sobre a Thread que deseja congelar e clique em "Freeze". Caso o seu foco seja a execução de uma só Thread, você poderá clicar com o botão direito e congelar todas as outras Threads com a opção "Freeze All Other Threads".

Pesquisa no código fonte

Para facilitar a localização de códigos fontes em seu projeto, foi melhorada a pesquisa, no Code Editor, ao teclar (Ctrl + F), uma barra de pesquisa será exibida e ao pesquisar, todas as correspondências serão destacadas na unit e na mesma barra, você notará botões para permitir a navegação entre os resultados.

Uma ferramenta completa de localização de código é encontrada no menu Search>Find in Files ou pela tecla de atalho Ctrl + Shift + F, como o nome da tela sugere, será possível localizar determinado conteúdo dentro dos arquivos, conforme mostra a **Figura 4**.

Podemos escolher em Where o tipo de pesquisa que será feita, onde será possível pesquisar em todos os arquivos do projeto atual (Search all files in project), todos os projetos do grupo (Search all files in Project group), todos os arquivos abertos pelo IDE (Search all open files) ou procurar em pastas (Search in directories).

Nota do DevMan

Threads é o termo que identifica um processamento paralelo, ou seja, a realização de uma tarefa durante a execução de outra. Multithread como o próprio nome sugere, realiza este processamento paralelo de uma maneira múltipla, criando vários processos simultâneos. É muito comum em Sistemas Operacionais, onde podemos realizar tarefas de forma simultânea e de maneira transparente. É importante comentarmos que Threads possuem um ciclo de vida, e devem ser manuseadas com muito cuidado, pois podem causar travamentos inesperados se não tratadas de forma correta.

Ao escolher pesquisar em pastas, informamos os tipos de arquivos a serem pesquisados em File mask (exemplo: *.pas) e informamos também as pastas que serão pesquisadas, separando-as por ponto e vírgula (exemplo: c:\temporario;c:\projetos\projeto1), ao selecionar o CheckBox "Include subdirectories" indicamos que todos os subdiretórios dos locais listados em Directories serão pesquisados e não somente a raiz.

Como opções de saída (output) tempos "Display results in separate tab", que criará uma aba em "Messages" para cada pesquisa efetuada e a opção "Group results by file" que cria uma árvore expansível para detalhar os resultados, agrupando-os pelo nome e localização do arquivo.

Liberdade durante a depuração

Ao depurar um código, sempre fazemos o uso de Breakpoints, porém muitas vezes o compilador não avalia o local definido por nós. Isto ocorre quando os colocamos em algum local errado, que não será checado. Mas a grande novidade é que a partir desta versão somos capazes de arrastar o Breakpoint para a linha de código que for necessária, mesmo durante o modo de depuração.

Na **Figura 5** vemos a marcação de um Breakpoint, que pode ser arrastado para outras partes do código, e o Current Statement (linha atual em que o cursor de Debug se encontra), que poderá ser arrastado para qualquer linha, desde que dentro do mesmo método. Ainda na **Figura 5**, se arrastarmos o cursor Current Statement ("flechinha") da posição 32 para 31, a próxima linha a ser executada será a 31.

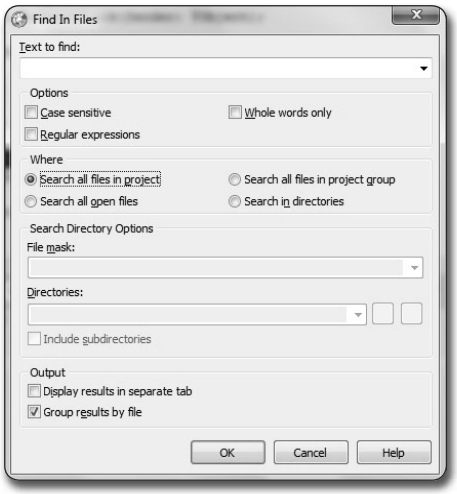


Figura 4. Tela Find In Files

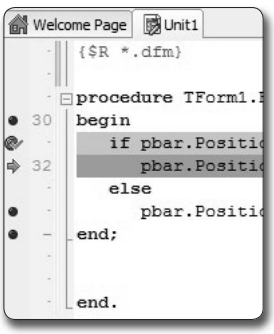


Figura 5. Cursores statement e Breakpoint

Autoformatação de código

A padronização e endentação são essenciais para facilitar o entendimento do código entre os membros da equipe de desenvolvimento e para que o código fique organizado é claro. Para isso temos a opção de auto-formatar (Ctrl + D), que instantaneamente formata o código. Para configurar esta opção acesse o menu Tools>Options>Formatter>Delphi. Para customizar o recurso de autoformatação do código fonte, há quatro subitens a se considerar sendo: "Indentation", "Spaces", "Line breaks" e "Capitalization".

Item Indentation (principais opções):

• Continuation indent: número adicional de espaços de endentação quando a expressão é continuada em outras linhas.

• Do not indent after position: determina a partir de qual posição o código não será mais endentado.
• Indent Begin and End keywords: Determina se begin e end serão identados.
• Indent blocks between Begin and End: Determina se os códigos entre begin e end serão endentados.

Item Spaces: Determina se haverá espaço antes, depois, antes e depois, ou nenhum espaço para algumas situações chaves, como para os sinais de comentários "//", "{" ou "(*".

Item Line breaks: Determina quando haverá quebras de linhas em determinadas situações, como por exemplo, após o begin, e regulamenta a quantidade de linhas em branco subsequentes no código.

Item Capitalization: Define as regras para letras maiúsculas e minúsculas de palavras reservadas, números hexadecimais e float, diretivas de compilação, para cada opção é permitido escolher entre as opções:
• As is (permanece como está, não altera o código);
• Lower case (todas as letras minúsculas);
• Upper case (todas as letras maiúsculas);
• Fist character In Upper Case (somente o primeiro caractere em maiúsculo);
• As first occurrence (sempre formata a palavra da maneira a qual foi escrita);

Use Unit Dialog

A utilização da janela Use Unit (Alt + F11) foi facilitada, há como filtrar para encontrar a Unit desejada, e ainda podemos escolher se será adicionada à sessão Implementation ou Interface. Lembrando que, em uses de Interface adicionamos as units que são referenciadas antes da implementação do código, por exemplo, se no Type possuir um TLabel, deverá declarar StdCtrls no uses da Interface, porém, se o seu objeto TLabel for declarado e criado somente em código implementado, poderá ser declarado em Implementation para ser criado dinamicamente. Veja um exemplo na **Figura 6.**

Driver nativo dbExpress para Firebird

A comunidade Firebird ganhou o tão pedido presente, junto com as melhorias no DBExpress Framework, foi desenvolvido um Driver de conexão nativa com o banco predileto da comunidade de desenvolvedores Delphi. Na **Figura 7** vemos o Data Explorer com a opção Firebird listada, e a partir desta janela, é possível acessar as Tabelas, Views e demais objetos existentes no banco de dados, tornando a integração Delphi/Firebird completa, sendo possível criar e manusear a conexão, criar e excluir tabelas ou alterar/criar campos no formulário ligado a uma tabela somente pelo Data Explorer, ou seja, todos os recursos nativos que estavam disponíveis para Interbase também estão disponíveis agora para Firebird.

Suporte a JSON

O Delphi 2010 trouxe o suporte ao JSON (JavaScript Object Notation ou Notação de Objetos JavaScript), que nada mais é do que um protocolo baseado em JavaScript muito utilizado para troca de dados, e possui uma notação muito simples de ser interpretada. É possível com este suporte a JSON no Delphi realizar transferência de objetos entre camadas do sistema em um ambiente multicamadas com o novo DataSnap, por exemplo. O JSON possui duas estruturas, uma é correspondente ao que chamamos no Delphi de objetos, e outra estrutura é o que chamamos de Arrays.

Além dos próprios recursos do Delphi, é possível encontrar classes para manipular e criar objetos JSON para Delphi, na sessão Links você poderá acessar o projeto Delphi Web Utils e o JSON SuperObject que suporta a conversão de XML para JSON, validador entre outras funcionalidades. A tendência é que o JSON se torne o possível substituto para o XML.

Novo DataSnap

O novo DataSnap atendeu uma das principais requisições dos usuários, que tratava da capacidade de transferir objetos entre camadas em aplicações multicamadas. Na versão 2009 do Delphi isso é possível, porém, somente para objetos do DBExpress. Com a ajuda do JSON, é possível a transferência de qualquer objeto com DataSnap entre aplicações cliente e

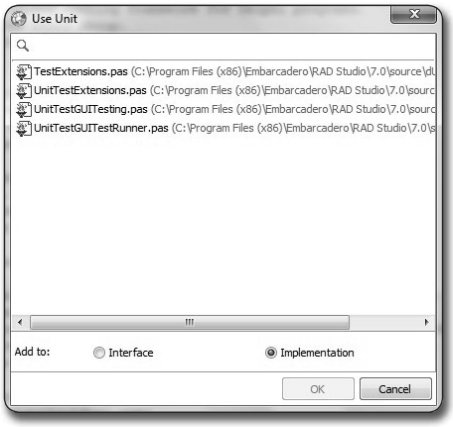


Figura 6. Janela Use Unit

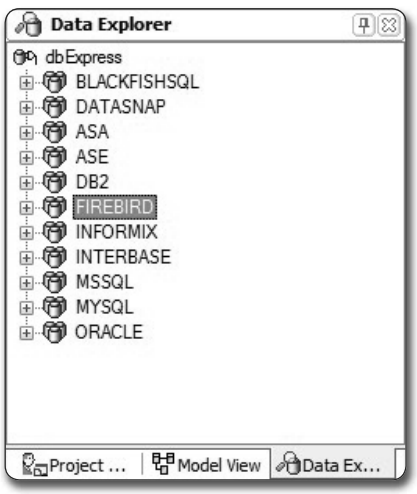


Figura 7. Suporte a Driver nativo do Firebird para DBExpress

servidora. Não se preocupe, não terá que reescrever seus objetos para JSON antes de enviá-los, há métodos para isso no Delphi 2010 e ao chamá-los seu objeto será automaticamente convertido em JSON em uma camada, e basta utilizar o método de conversão na outra camada que receberá os objetos para transcrever de JSON para um tipo comum novamente.

O suporte a Callbacks, outra grande novidade do novo DataSnap, possibilita que uma aplicação servidora antes mesmo de finalizar um procedimento, consiga trocar mensagens/informações com a camada cliente, podendo realizar uma maior interação entre as camadas e obter informações de ambos os lados, permitindo criar soluções bastante criativas em suas aplicações.

Também foi criado um novo Wizard que auxilia no desenvolvimento de uma aplicação que utiliza DataSnap. Você pode acessá-lo

através do menu File>New>Other>Delphi Projects>DataSnap Server e selecione a opção DataSnapServer do lado direito, você notará uma nova tela para lhe auxiliar na criação, como mostra a **Figura 8**.

Delphi Natural Input

Há muito tempo esperado pelos desenvolvedores Delphi, os recursos de Natural Input vieram como uma grande novidade, que deve ser aproveitada ao máximo por nós. Sendo ela uma nova tecnologia de entrada de dados e interação com o usuário baseada em toques. Isso mesmo, toques! Lembra-se do filme Minority Report? Onde simplesmente arrastavam as janelas com os dedos? Da tecnologia do iPhone e Microsoft Surface em que o usuário deixa de utilizar pequenos botões para arrastar os dedos na tela, e de acordo com o sentido do movimento o programa interpretar o que deve ser feito? Completamente possível e prático no Delphi.

Esta tecnologia pode ser encontrada na paleta de componentes Gestures, tendo nela um Manager, um visualizador de gestos, um objeto para listar os gestos, e um para componente para gravação, e realmente estes quatro objetos são suficientes para implementar qualquer tecnologia de Natural Input disponibilizada pelo Delphi.

As tecnologias de Delphi Natural Input são divididas em três tópicos sendo: Touch básico, MultiTouch e Gestures. Touch básico permite que os dedos substituam o botão direito do mouse e é ideal para aplicações com botões grandes ou assistentes, pode ser de grande valia em quiosques, pontos de venda e autoatendimento por exemplo. MultiTouch estende as capacidades do Touch, reconhecendo mais de um toque de uma só vez, permitindo, por exemplo, o redimensionamento de imagens utilizando dois dedos em suas extremidades assim como a rotação e zoom, porém, é suportado somente pelo Windows 7 e hardwares mais atuais. Gestures, a terceira tecnologia, atua com o movimento do dedo ou mouse, disparando um evento ao reconhecer um movimento (gesto). Existem mais de 30 gestos padrões que são reconhecidos, bastando apenas atribuir ações aos mesmos para ativá-los, além disso, é permitido criar seus próprios gestos. Esta tecnologia é suportada em quase todos os sistemas Touch. Embora o MultiTouch seja suportado somente pelo Windows 7, Touch e Gestures são suportados pelas versões Windows XP ou posteriores.

Todo componente visual agora possui a propriedade Touch, à qual você atribui o componente Gesture Manager, podendo personalizar ações para quando o mouse for movimentado sobre o objeto (independente de cliques) ou então, quando for realizado o próprio movimento dos dedos do usuário. E esta atribuição pode ser feita sem a digitação de uma linha de comando, somente configurando as propriedades do Touch do objeto.

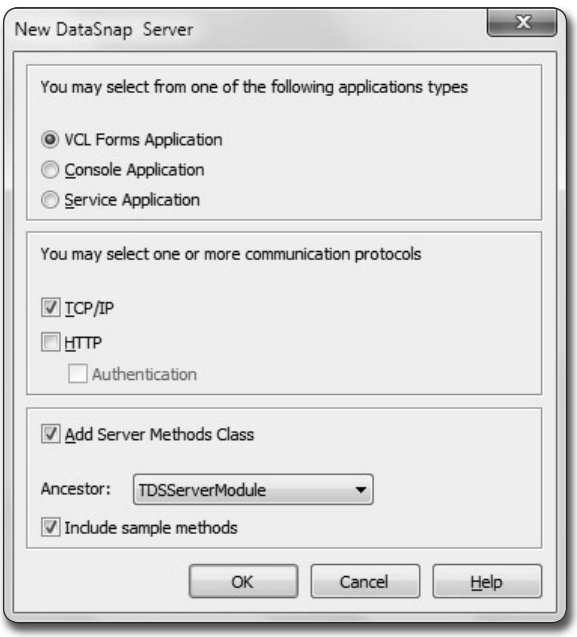


Figura 8. Wizard de criação de um DataSnap Server

Figura 9. Teclado virtual em layout Standard

Componente Teclado Virtual

Para facilitar a entrada de texto via Touch, foi disponibilizado o componente TTouchKeyboard para a criação de um teclado virtual. O teclado muda seu layout de acordo com as configurações de linguagem definidas no painel de controle do Windows, facilitando a interação do usuário com o mesmo. Além desta facilidade, é permitido alterar os Captions das teclas.

Há dois modos do teclado que são definidos em sua propriedade layout, Standard (completo) ou NumPad (teclado numérico). Na **Figura 9** podemos conferir o teclado virtual em seu modo Standard e na **Figura 10** o mesmo componente em modo NumPad.

Figura 10. Teclado virtual em layout NumPad

O que nos espera nas próximas versões

A Embarcadero conta com mais de 1.5 milhões de clientes em todo o mundo, e é pensando em seu crescimento e em satisfazer os clientes atuais que já vem desenvolvendo novos recursos para a próxima versão do Delphi em dois projetos pilotos, o projeto Delphi X e o projeto Commodore.

No projeto Delphi X está sendo desenvolvido o Cross-platform Windows, MacOS e Linux, com isso, poderemos desenvolver em Windows e compilar para estas outras plataformas, assim como criar Server e Clients para Windows, MacOS e Linux. Para tanto, está sendo desenvolvida a Cross-platform Component Library.

No projeto Commodore está sendo desenvolvido o compilador nativo para 64-bits, com suporte a aplicações Multi Core/Multi Threaded e paralelismo na RTL.

A tendência continua em facilitar o desenvolvimento, deixando a IDE mais leve e cada vez mais compatível com os hardwares mais atuais.

Estamos seguros que o Delphi irá crescer cada vez no que depender da Embarcadero, a nós, compete solicitar cada vez mais melhorias e aprender as tecnologias novas para que possamos tirar proveito máximo de todos estes recursos e novidades que são disponibilizados a cada versão.

Conclusão

Sempre devemos conhecer o IDE por completo para que nosso desenvolvimento seja rápido e dinâmico, e vimos que a Embarcadero se preocupou com esse dinamismo. A navegação e usabilidade se tornaram muito mais simples com as novas opções disponíveis, e com o uso do IDE Insight podemos realizar todas as tarefas cotidianas de uma maneira mais ágil, permitindo um desenvolvimento mais produtivo e um conhecimento maior da IDE. O recurso de busca aprimorada para pesquisar em pastas nos permite que trabalhemos mais no próprio IDE, deixando de utilizar as pesquisas do assistente do Windows para localizar blocos de códigos fontes que necessitamos em outros projetos por exemplo. A liberdade de determinar, alterar Breakpoints em modo de depuração e depurar/congelar Threads nos coloca no controle definitivo da execução. Tudo isso torna o dia-a-dia do desenvolvedor mais produtivo, marca registrada do Delphi nos seus mais de 15 anos de idade.

Álison Bissoli Dias
bissolidias@gmail.com
http://garotodesktop.blogspot.com
Atualmente é analista de sistemas na Mastermaq Softwares. Especialista em ERP, Supply Chain, CRM, gestão de requisitos, gestão de processos e implantações de sistemas. Formado em Processamento de Dados em 2004 pela Escola Técnica Juscelino Kubtischeck e cursando o curso superior tecnológico em Análise e Desenvolvimento de Sistemas pela Universidade Metodista de Minas.

GlyFx
http://www.glyfx.com/embarcadero/

Delphi Web Utils
http://sourceforge.net/projects/is-webstart/

JSON SuperObject
http://www.progdigy.com/?page_id=6

Blog do autor
http://garotodesktop.blogspot.com

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Cursos Online

Web Mobile PLUS

A Revista **WebMobile** oferece para seus assinantes uma série de Cursos Online de alto padrão de qualidade .

Conheça abaixo os cursos já disponíveis:
www.devmedia.com.br/curso/webmobile

- Curso em .NET: **Aplicação completa de orçamento Doméstico no Visual Studio 2005**
- Curso em Java: **Introdução ao desenvolvimento para celulares com J2ME**

A sua melhor opção de aprendizagem!
Assine a **WebMobile** e
Comece já seu treinamento!
www.devmedia.com.br/assine

DevMedia group Mais Informações: www.devmedia.com.br/central - Tel.: 21 3382-5038

Delphi Prism

Acesso nativo ao Firebird

Nós desenvolvedores Delphi Win32 ou mesmo Web temos o hábito de associar o Delphi diretamente com Firebird, dito como "feitos um para o outro". Infelizmente o Firebird não é a melhor escolha para o ambiente Web devido ao seu protocolo (embora melhorado, ainda pesado), porém, há quem goste de usá-lo em qualquer ocasião, ou mesmo em casos onde se deseja portar uma parte de um projeto já pronto, disponibilizando módulos deste mesmo projeto para a Web. Com certeza se você deseja trabalhar com .NET de alguma forma, utilizando o Delphi Prism ou qualquer outra linguagem, deve saber que o banco de dados que melhor se adapta ao .NET é o Microsoft SQL Server.

Para quem já desenvolve, ou ao menos teve um primeiro contato com a tecnologia ADO.NET do .NET Framework, já deve ter notado a grande facilidade que é trabalhar diretamente com dados utilizando esta tecnologia. Visto que já temos conhecimento de acesso a dados no Delphi através do dbExpress por exemplo, utilizar o ADO.NET torna-se uma tarefa muito simples e digamos até muito parecida. Contudo, você perceberá que não é possível utilizá-los de forma nativa na instalação padrão do Visual Studio e Delphi Prism, mas isto não significa que não podemos realizar esta tarefa.

Para trabalharmos com o Firebird em conjunto com ADO.NET utilizando o Delphi Prism necessitaremos do Firebird Data Provider, que nada mais é (como o próprio nome sugere) o provedor de conexão nativa com o Firebird, assim será possível utilizá-lo no ADO.NET de forma nativa, mas ainda teremos uma restrição. Não será possível criar conexões e DataBinding com os componentes utilizando as SmartTags (setas localizadas no canto superior direito dos componentes no VS), ou mesmo utilizar-se de assistentes para efetuar o processo de conexão. A razão para tudo isso é que devemos possuir o DDEX do Firebird instalado no .NET para assim conseguirmos o acesso em tempo de Design.

Neste artigo veremos como proceder com a instalação do Firebird Data Provider, e do DDEX for Firebird, conheceremos as duas maneiras de utilizar o Firebird com o ADO.NET adotando para isso a forma tradicional via linha de comando, ou mesmo através dos assistentes de conexão e os processos de DataBinding realizados através do Wizard. Então mãos à obra.

Resumo DevMan

De que trata o artigo:

Neste artigo mostraremos como utilizar ADO.NET com o Firebird de forma nativa, ilustrando para o leitor como trabalhar com o Firebird no Delphi Prism tanto em linhas de comando quanto da forma mais simples, utilizando-se de assistentes do Visual Studio, fazendo uso do DDEX e do Firebird Data Provider.

Para que serve:

Como já é do nosso conhecimento, acessar bancos de dados de maneira nativa sempre é mais rápido do que outras formas de acesso. Justamente para isso utilizaremos o Firebird Data Provider que nos possibilita esta tarefa, o DDEX nos permite trabalhar com o ADO.NET e Firebird em modo visual, ou tempo de Design.

Em que situação o tema é útil:

Na maioria das aplicações com Delphi Prism e .NET como um todo, o banco utilizado é o SQL Server. No entanto, para quem já possui um sistema Desktop em Delphi Win32, certamente será mais fácil reaproveitar o seu banco Firebird para criar uma versão Web do aplicativo, por exemplo, usando o Delphi Prism. Nesse caso, o acesso nativo é a melhor escolha.

Delphi Prism:

Desde o lançamento do Delphi Prism, a ClubeDelphi vem mostrando uma série de artigos que abordam a utilização e desenvolvimento de projetos utilizando o IDE do Visual Studio para a produção de aplicações .NET. Neste artigo iremos mais uma vez tratar do Delphi Prism, mostrando desta vez como utilizar o Firebird de forma nativa em suas aplicações Web/Win32, fazendo uso do Firebird Data Provider e do DDEX, que permite a utilização do ADO.NET de forma visual, usando assistentes. No decorrer do artigo também veremos dicas de todo o processo, para tornar esta tarefa o mais simples possível.

Nota

Para o exemplo foi utilizado o Firebird 2.1. Na sessão Links você encontra o endereço para download do mesmo.

Instalando o Firebird Data Provider

Vá ao site oficial do Firebird (indicado na sessão Links) e acesse o menu Downloads conforme mostra a **Figura 1**. Em seguida, faça o download dos arquivos circulados na **Figura 2**.

Primeiramente atentaremos a instalação do Firebird Data Client que refere-se ao primeiro círculo da **Figura 2.** A

Nota do DevMan

Utilizando tradicionalmente o Delphi podemos notar a presença da paleta Data Controls. Esta paleta diz respeito aos componentes que fazem acesso direto ao banco de dados, funcionando através de Data Sources e Data Fields. Em .NET não há com o que se preocupar, pois todos os controles possuem a possibilidade de exibir dados diretamente de uma fonte de dados, e a esta característica damos o nome de DataBinding, ou seja, nada mais é do que o vínculo de qualquer controle com uma fonte de dados.

Nota do DevMan

DDEX é a sigla para Data Design Extensibility, ou extensibilidade de dados em Design, como o próprio nome já sugere, nos permite trabalhar com acesso a dados diretamente em tempo de Design, permitindo utilizar SmartTags ou recursos de Wizards disponíveis no IDE do VS.

instalação é muito simples, tratando-se do tradicional "Next, Next e Finish". Atente apenas para o local em que serão extraídos os arquivos, como mostra a **Figura 3**.

Utilizando ADO.NET e Firebird via linha de comando

A partir daí já poderemos desenvolver no Delphi Prism utilizando o ADO.NET com Firebird, porém apenas através de linhas de comando, necessitando apenas referenciar o assembly da dll, para isso crie um novo projeto ASP.NET clicando em File>New>Web Site e configure seu projeto como ilustra a **Figura 4**.

Note na **Figura 4** que definimos em Location o "modo de funcionamento" do nosso Web Site, escolhendo File System teremos uma aplicação rodando em um servidor ASP.NET local. Em Language podemos configurar qual será a linguagem utilizada no desenvolvimento, neste caso deixe selecionado Oxygene, para

Figura 1. Página oficial do Firebird

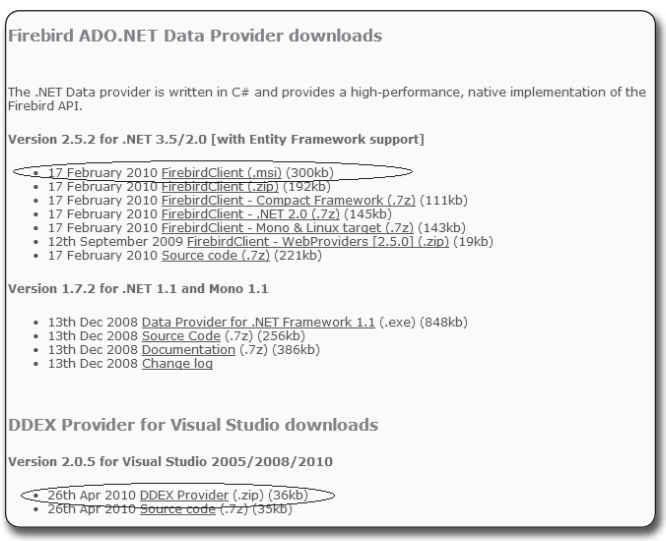


Figura 2. Downloads dos arquivos necessários

utilizarmos o Delphi Prism e por último clique no botão OK.

Assim que a página for criada, direcione-se ao Solution Explorer e clique de direita, selecionando a opção Add Reference conforme mostra a **Figura 5.**

Após ter selecionado a opção Add Reference, uma nova janela será exibida, ilustrando as referências já utilizadas no seu projeto. Selecione a guia Browse e procure pelo diretório onde instalamos o Firebird Data Provider. Lembre-se que por padrão ele estará em C:\Arquivos de Programas\FirebirdClient. Selecio-

ne o arquivo "FirebirdSql.Data.FirebirdClient.dll" para incluir a referência do provider ao seu projeto, como mostra a **Figura 6**.

A partir daqui podemos utilizar o Firebird com ADO.NET normalmente, realizando operações CRUD por exemplo, apenas adicionando o Namespace "Firebirdsql.Data.FirebirdClient" como o exemplo a seguir:

```
namespace;

interface

uses
  System,
  System.Data,
  System.Configuration,
  System.Web,
  System.Web.Security,
  System.Web.UI,
  System.Web.UI.WebControls,
  System.Web.UI.WebControls.WebParts,
  System.Web.UI.HtmlControls,
  FirebirdSql.Data.FirebirdClient;
```

Selecione a guia Design e adicione um GridView para fazermos um pequeno teste. Dê um duplo clique sobre a página e no evento Page_Load implemente a codificação da **Listagem 1.**

Note na **Listagem 1** que criamos cinco variáveis. A variável StringConexao, como o próprio nome sugere, recebe o conteúdo da String de conexão com o

Nota do DevMan

CRUD é o que chamaríamos de as quatro operações básicas de manipulação a dados. Create (inserção), Read (seleção) Update (atualização) e Delete (exclusão).

Nota

Durante todo o processo de instalação do DDEX o Visual Studio deve estar fechado.

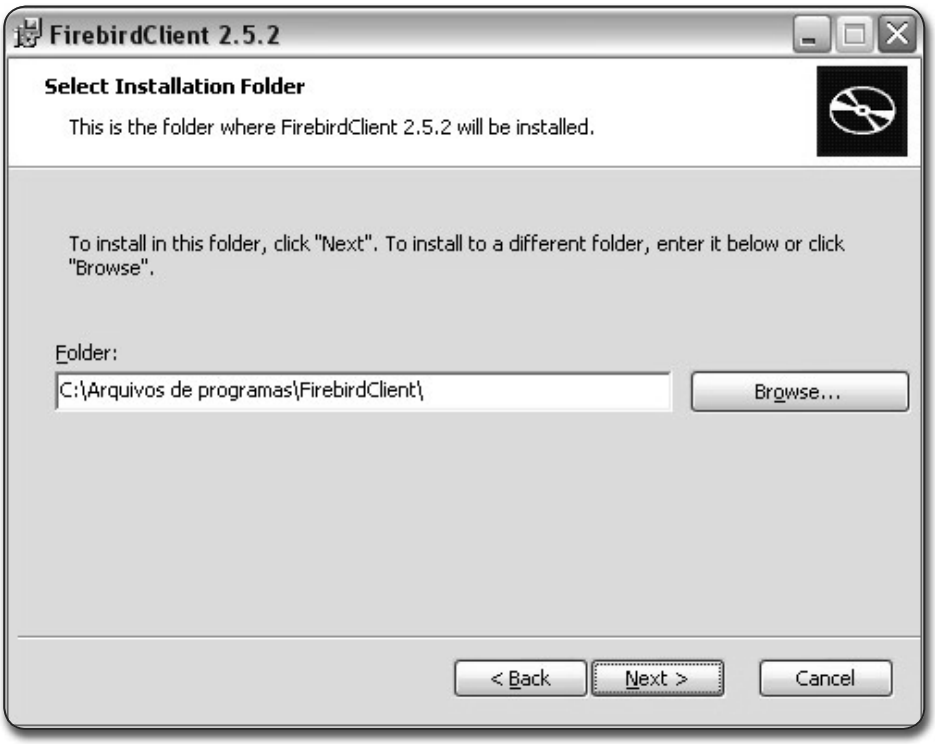


Figura 3. Local de extração dos arquivos

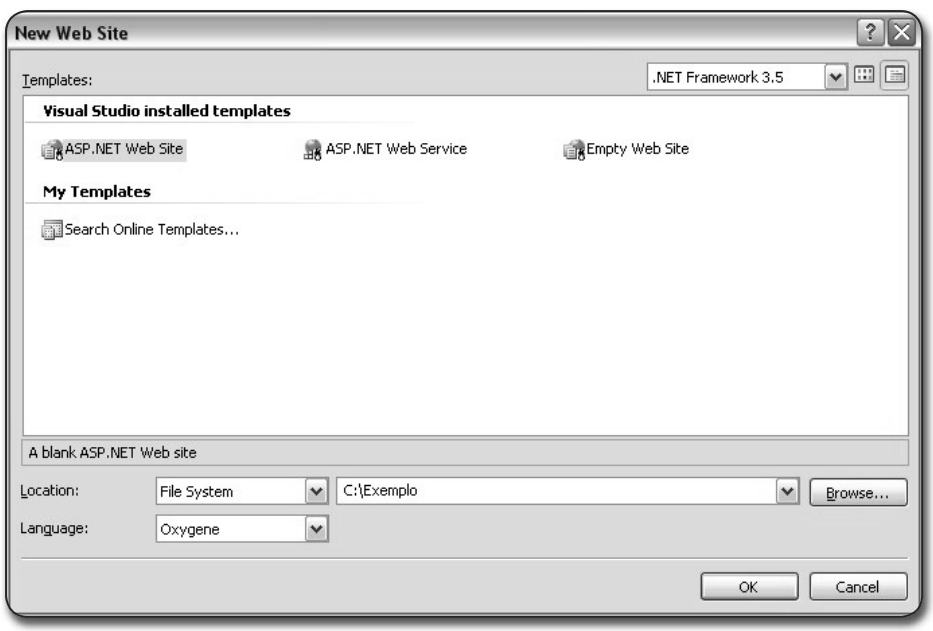


Figura 4. Configurando o Web Site

Listagem 1. Utilizando ADO.NET e Firebird nativo via ADO.NET e linhas de comando

```
method _Default.Page_Load(sender: Object; e: EventArgs);
begin
   var StringConexao := "User=SYSDBA;Password=masterkey;Database=
   C:\Arquivos de programas\Firebird\Firebird_2_1\examples\empbuild\
    Employee.fdb;DataSource=localhost;";
   var Con := new FbConnection(StringConexao);
   var sql := "SELECT COUNTRY, CURRENCY FROM COUNTRY";
   var DataAdapter := new FbDataAdapter(sql,Con);
   var Ds := new DataSet();
   DataAdapter.Fill(Ds);
   GridView1.DataSource := Ds;
   GridView1.DataBind();
end;
```

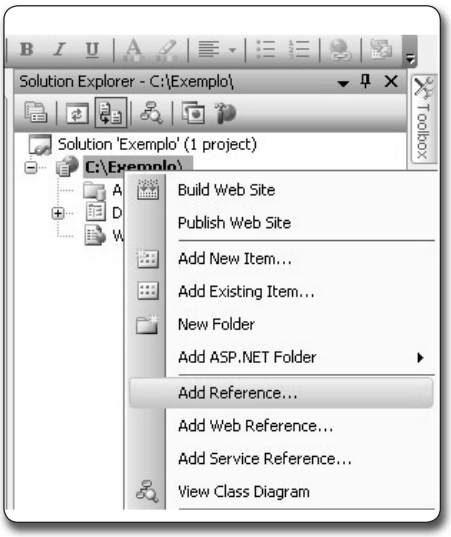


Figura 5. Adicionando uma referência ao projeto

banco de dados, ilustrando o acesso ao banco Employee. A seguir criamos um objeto FBConnection que receberá esta String de conexão, uma variável sql que contém a instrução SQL para acesso a tabela Country, um DataAdapter, que nada mais é do que um "intermediador", fazendo um papel parecido com o do nosso DataSetProvider no Delphi Win32. A seguir o DataAdapter através do método Fill preenche um DataSet, que aqui é uma fonte de dados. O que dizemos ao GridView aqui é: a sua fonte de dados (Data Source) será o objeto Ds, e realize em seguida o processo de DataBind, que como o DevMan já citou, é o processo de conectar o controle à fonte de dados. Execute sua aplicação e note o resultado na **Figura 7.**

A desvantagem aqui é que não conseguimos utilizar os Wizards nativos do VS, o que torna nosso trabalho mais "improdutivo", visto que os Wizards nos facilitam e agilizam muito no desenvolvimento. A seguir vamos resolver esse pequeno impasse utilizando o DDEX.

Instalando o Firebird DDEX

O Firebird DDEX como dito anteriormente permite a configuração do Firebird em conjunto com o ADO.NET no VS de modo visual, facilitando muito a criação de conexão e o processo de DataBinding com os objetos. O primeiro passo é realizar o download (disponível

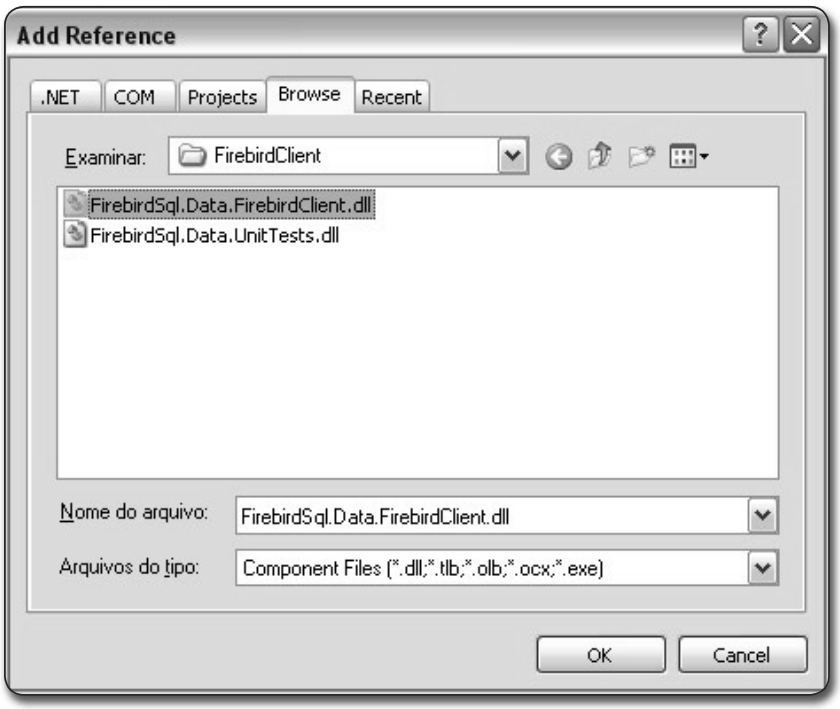
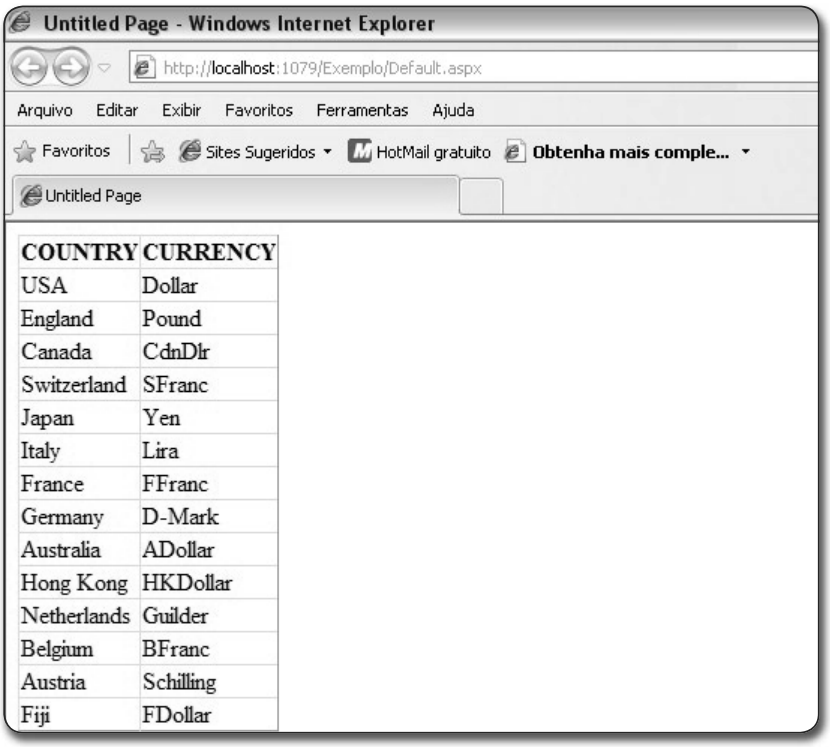


Figura 6. Adicionando a referência do Data Provider ao projeto



COUNTRY	CURRENCY
USA	Dollar
England	Pound
Canada	CdnDlr
Switzerland	SFranc
Japan	Yen
Italy	Lira
France	FFranc
Germany	D-Mark
Australia	ADollar
Hong Kong	HKDollar
Netherlands	Guilder
Belgium	BFranc
Austria	Schilling
Fiji	FDollar

Figura 7. GridView utilizando ADO.NET e Firebird

na sessão Links) e extrair para uma pasta de sua preferência conforme mostra a **Figura 8**.

Note na **Figura 8** que criamos uma pasta em C:\Arquivos de Programas\Firebird DDEX Provider e extraímos todo o conteúdo do arquivo .rar para esta pasta. O próximo passo é utilizar o utilitário do gacutil do VS para registrar as .dll em seu próprio assembly. Para isso acesse o menu Iniciar e procure dentro da pasta do VS a pasta do Visual Studio Tools, acessando em seguida o Visual Studio 2008 Command Prompt. Via Command acesse a pasta em que você extraiu seus arquivos e em seguida digite a seguinte linha de comando:

```
gacutil /i FirebirdSql.VisualStudio.DataTools.dll
```

Observe na **Figura 9** que você receberá uma mensagem de que a dll foi adicionada com êxito.

O próximo passo é alterarmos as informações da .dll no arquivo **machine.config**, que é responsável por carregar todos os assemblys durante o processo de inicialização do IDE. Para isso acesse o diretório "C:\Windows\Microsoft.NET\Framework\v2.0.50727\CONFIG". Antes de alterarmos é ideal realizarmos uma cópia de segurança deste arquivo, então simplesmente copie, cole e renomeie o arquivo machine.config para darmos início às alterações necessárias.

Abra o arquivo machine.config e logo no início na sessão **<configSections>** do xml vamos inserir a seguinte linha:

```
<section name="firebirdsql.data.firebirdclient"
type="System.Data.Common.DbProviderConfiguratio-
nHandler, System.Data, Version=%RuntimeVersion%,
Culture=neutral, PublicKeyToken=Token" />
```

Observe que na linha anterior as cláusulas Version, Culture e PublicKeyToken estão em negrito. Isto é porque temos que nos atentar principalmente a estas informações, informando-as corretamente. Para isto vamos acessar o diretório "C:\Windows\Assembly", e procurar pelos assemblys do Firebird conforme mostra a **Figura 10.**

Observando a **Figura 10** podemos notar as informações que estamos procurando para preencher nosso xml. Neste caso informaremos conforme o ilustrado. É importante saber que estes valores podem variar em cada máquina. Após ter obtido as informações necessárias, aquela linha que adicionamos ficou da seguinte maneira:

Figura 8. Diretório do DDEX

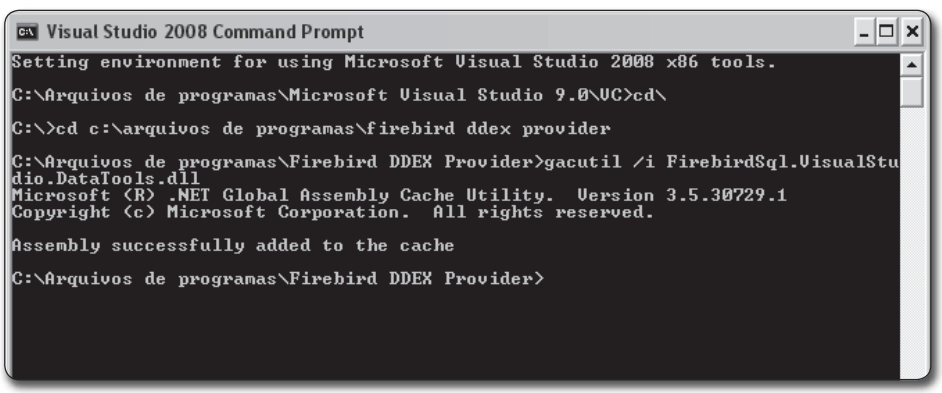


Figura 9. Registrando a dll na cache através do gacutil

Figura 10. Informações do assembly

```
<section name="firebirdsql.data.firebirdclient"
type="System.Data.Common.DbProviderConfiguratio-
nHandler, System.Data, Version=2.5.2.0, Culture=neutral,
PublicKeyToken=3750abcc3150b00c" />
```

Ainda no machine.config procure por **<DbProviderFactories>** e insira a seguinte informação:

<add name="FirebirdClient Data Provider" invariant="FirebirdSql.Data.FirebirdClient" description=".Net Framework Data Provider for Firebird" type="FirebirdSql.Data.FirebirdClient.FirebirdClientFactory, FirebirdSql.Data.FirebirdClient, Version=**%Version%,** Culture=**%Culture%,** PublicKeyToken=**%PublicKeyToken%"** />

Observe que novamente foram destacadas as informações pertinentes a nós. Novamente de acordo com a **Figura 10** iremos incluir estas informações, deixando esta linha como a seguir:

<add name="FirebirdClient Data Provider" invariant="FirebirdSql.Data.FirebirdClient" description=".Net Framework Data Provider for Firebird" type="FirebirdSql.Data.FirebirdClient.FirebirdClientFactory, FirebirdSql.Data.FirebirdClient, Version=**2.5.2.0**, Culture=**neutral**, PublicKeyToken=**3750abcc3150b00c**" />

Salve o arquivo machine.config para passarmos para a última etapa de configuração. Sendo assim novamente vá ao diretório em que você extraiu os arquivos do DDEX e procure pela pasta "reg_files". Dentro deste diretório você encontrará pastas conforme a versão do VS que você está utilizando, que no meu caso é a versão 2008. Entre dentro deste diretório e note que temos dois arquivos de registro, denominados de: "FirebirdDDEXProvider32.reg" e "FirebirdDDEXProvider64.reg", além da pasta withSDK. É justamente com estes arquivos que vamos trabalhar. Estes arquivos irão registrar nosso DDEX no registro do Windows, porém, é importante atentar aqui para a arquitetura que você está utilizando, se corresponde ao 32 ou 64 bits. No meu caso irei trabalhar no arquivo "FirebirdDDEXProvider32.reg", por estar utilizando uma versão 32 bits.

Abra o arquivo correspondente a sua versão utilizando um editor de sua preferência, no meu caso realizarei as modificações através do bloco de notas. Observe a seguinte informação do registro na **Listagem 2**.

Novamente na **Listagem 2** temos a linha CodeBase em negrito. Isto porque esta informação deverá ser alterada para o diretório em que se encontra a .dll referenciada na **Listagem 2** ("FirebirdSql.VisualStudio.DataTools.dll"). Aqui temos uma observação a fazer. Conforme você deve ter notado após a instrução %**Path%**, temos duas "\\" para referenciar o diretório. Este é o grande detalhe. Onde for marcado como um diretório, deverá obrigatoriamente ser digitado no lugar de uma "\", duas "\\". Portanto nossa informação ficará como na **Listagem 3.**

Após ter modificado, salve o arquivo e execute-o em seguida, clicando em SIM quando for questionado se deseja adicionar estas informações ao registro. Novamente abra o Visual Studio e crie um novo projeto, acessando File>New>Web Site e configurando-o como mostra a **Figura 11.**

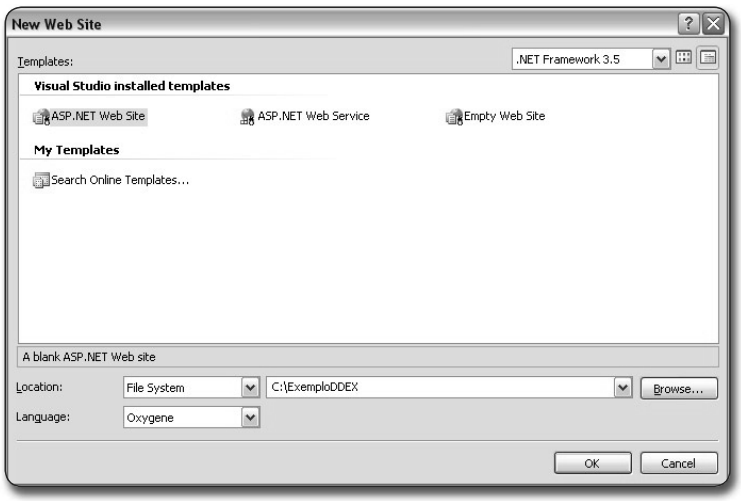


Figura 11. Configurando um novo Web Site

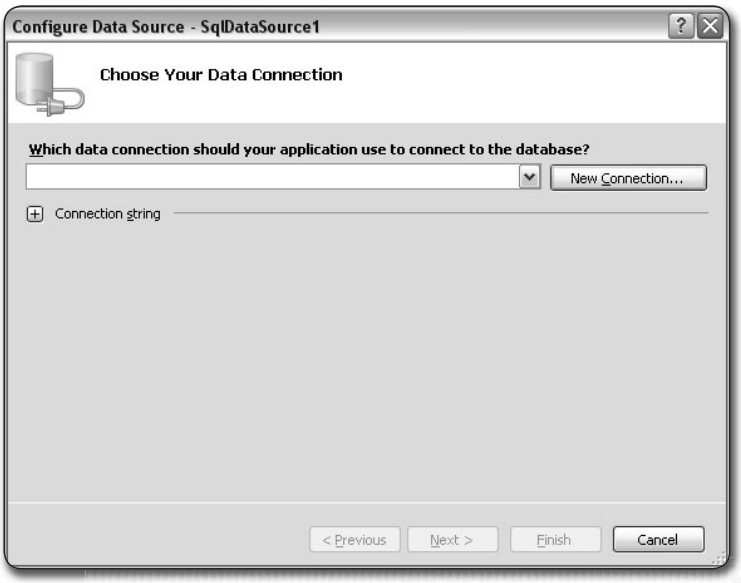


Figura 12. Criando uma nova conexão

Nota

Caso ao clicar nesta janela para digitar algo esta tela seja automaticamente fechada, alguma informação no machine.config pode estar com problemas. Neste caso feche o IDE, abra o arquivo novamente, faça as devidas modificações, salve-o e abra novamente o VS.

Nota

A instalação do DDEX não funciona na versão Express do Visual Studio.

Após o carregamento do projeto selecione a guia Design e novamente insira dentro da div um GridView (paleta Data). Clique na SmartTag (seta no canto superior direito) do controle e selecione em Choose Data Source a opção New Data Source. Na próxima janela selecione a fonte de dados como sendo um Database e clique em OK. Na próxima janela selecione a opção New Connection, para informarmos a conexão de string para acesso ao banco de dados (**Figura 12**). Perceba que na próxima janela que nos pede para informar o Data Source agora temos a opção Firebird Data Source (**Figura 13**). Selecione esta opção e clique em Continue.

Perceba que agora é possível configurarmos de modo visual o acesso ao banco de dados. Configure o banco de dados de exemplo conforme a **Figura 14** e clique no botão OK.

Após ter clicado em OK você voltará automaticamente para a tela de seleção da conexão de string, porém desta vez já estará preenchida com o Employee. Na próxima tela você será questionado se deseja adicionar estas informações no arquivo web.config, mantenha tudo como está e apenas clique em Next. Note agora que você já possui acesso às tabelas do banco de dados e demais campos conforme você pode notar na **Figura 15**. Selecione os campos Country e Currency e clique em Next. A última tela do assistente permite você testar se sua query está funcionando. Deste modo clique em Test Query e observe o resultado na **Figura 16.**

Após ter finalizado o assistente execute seu WebSite pressionando Ctrl+F5

Listagem 2. Informações de registro contidas no arquivo .reg

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VisualStudio\9.0\DataProviders\{92421248-F044-483A-8237-
74C7FBC62971}]
@=".NET Framework Data Provider for Firebird"
"DisplayName"="Provider_DisplayName, FirebirdSql.VisualStudio.DataTools.Properties.Resources"
"ShortDisplayName"="Provider_ShortDisplayName,FirebirdSql.VisualStudio.DataTools.Properties.Resources"
"Description"="Provider_Description, FirebirdSql.VisualStudio.DataTools.Properties.Resources"
"CodeBase"="%Path%\\FirebirdSql.VisualStudio.DataTools.dll"
"InvariantName"="FirebirdSql.Data.FirebirdClient"
"Technology"="{77AB9A9D-78B9-4ba7-91AC-873F5338F1D2}"
```

Listagem 3. Preenchendo o Path no registro

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VisualStudio\9.0\DataProviders\
 {92421248-F044-483A-8237-74C7FBC62971}]
@=".NET Framework Data Provider for Firebird"
"DisplayName"="Provider_DisplayName, FirebirdSql.VisualStudio.DataTools.Properties.Resources"
"ShortDisplayName"="Provider_ShortDisplayName,FirebirdSql.VisualStudio.DataTools.Properties.Resources"
"Description"="Provider_Description, FirebirdSql.VisualStudio.DataTools.Properties.Resources"
"CodeBase"="C:\\Arquivos de programas\\Firebird DDEX Provider\\FirebirdSql.VisualStudio.DataTools.dll"
"InvariantName"="FirebirdSql.Data.FirebirdClient"
"Technology"="{77AB9A9D-78B9-4ba7-91AC-873F5338F1D2}"
```

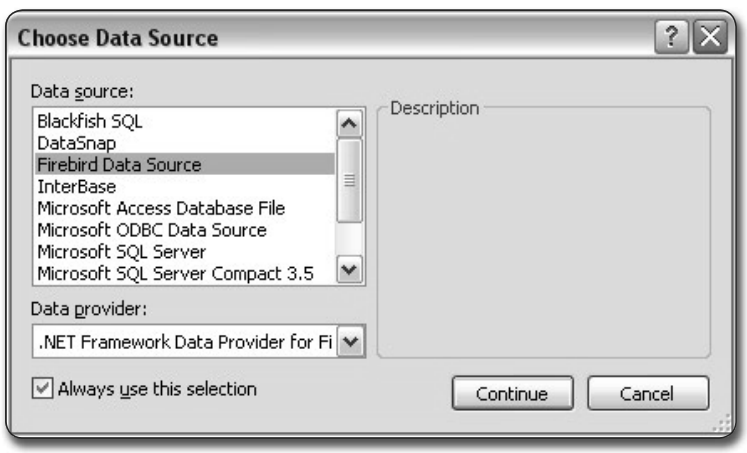


Figura 13. Data Source do Firebird

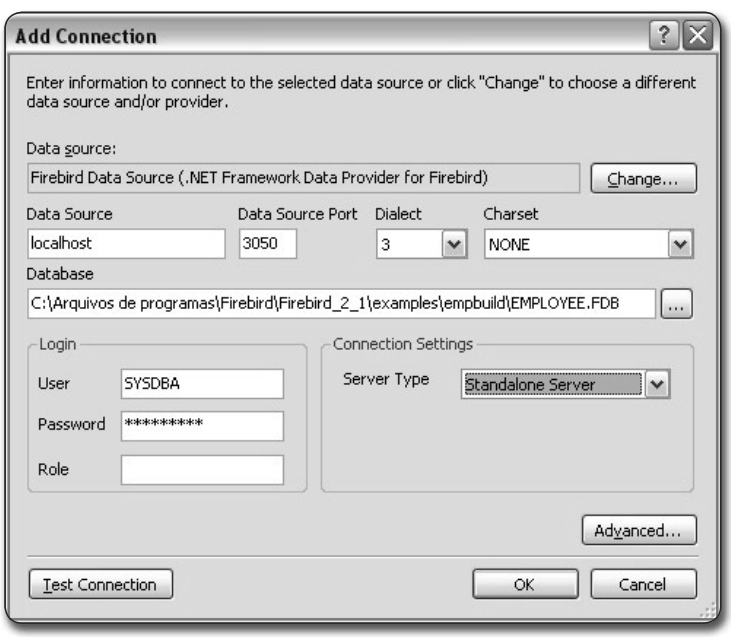


Figura 14. Configurando o acesso ao banco de dados em modo visual

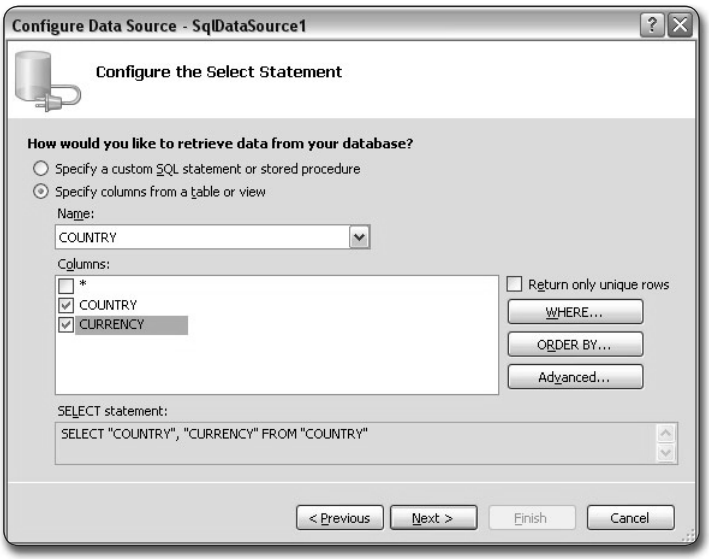


Figura 15. Selecionando a tabela e campos do banco de dados

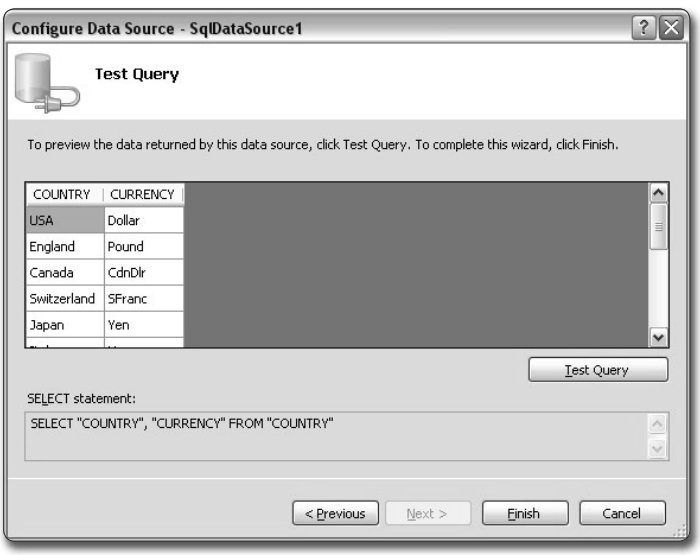


Figura 16. Teste da Query

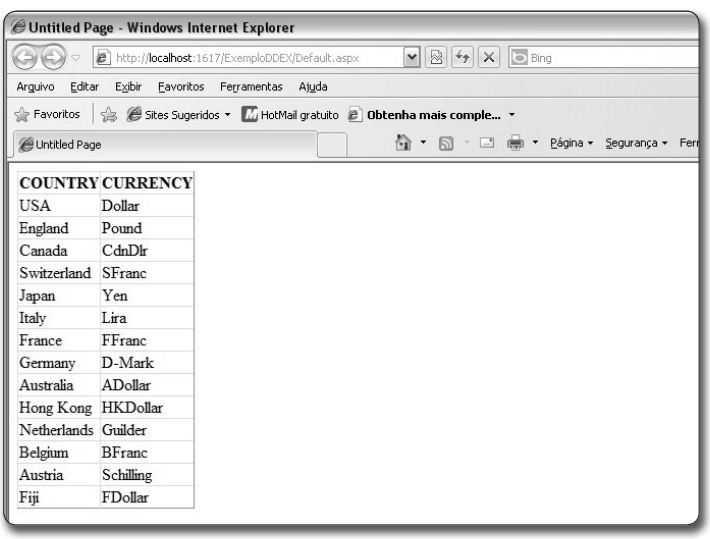


Figura 17. GridView em funcionamento com o DDEX e ADO.NET

e observe o resultado final, na **Figura 17** que é o mesmo da **Figura 7.**

Conclusão

Percebeu o quanto é produtivo utilizar os assistentes de conexão do ADO.NET no VS? Todo o código que podemos ver na listagem foi automaticamente realizado para nós sem que tivéssemos que nos preocupar. Da forma como vimos, foi possível compreendermos a grande diferença entre o Data Provider e o DDEX, possibilitando a visualização do mesmo resultado no final das contas, porém de uma maneira muito mais ágil. Utilizar o .NET com o Delphi Prism é como utilizar o Delphi Win32 e agora podemos trabalhar com o Firebird de um modo muito parecido com o dbExpress por exemplo. Um grande abraço e até a próxima!

Daniel Sobrinho Laporte

daniel.sobrinho@devmedia.com.br
daniel.laporte@gmail.com
www.twitter.com/daniel_laporte

Colunista e editor técnico da Revista ClubeDelphi. Graduado em Sistemas de informação e pós-graduado em Administração de Bancos de Dados pela Universidade de Franca. É analista de sistemas e desenvolvedor Delphi desde 2005 desenvolvendo aplicações cliente/servidor e multicamadas.

Página oficial do Firebird
http://www.firebirdsql.org

Download Firebird 2.1
http://sourceforge.net/projects/firebird/files/firebird-win32/2.1.3-Release/Firebird-2.1.3.18185_0_Win32.exe/download

Download Firebird DDEX
http://sourceforge.net/projects/firebird/files/firebird-net-provider/DDEX%202.0.5/FirebirdDDEXProvider-2.0.5.zip/download

Download Firebird Data Provider
http://sourceforge.net/projects/firebird/files/firebird-net-provider/2.5.2/NETProvider-2.5.2.msi/download

Blogs do autor
http://twitter.com/daniel_laporte

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Mapeamento Objeto /

Desenvolva aplicações de BD usando poder

Boa parte dos desenvolvedores Delphi se entusiasma com a teoria que envolve a orientação a objetos e toda sua boa prática, podendo citar dentre elas os Design Patterns. Começam a modelar seus sistemas, suas regras de negócio, porém, quando vão construir sua interface de usuário já se deparam com uma dificuldade muito comum. Como ligar os dados dos objetos aos controles visuais e vice versa? Depois, outra dúvida aparece, como vou salvar meus objetos? Minha classe irá se tornar uma tabela? Como manter o relacionamento que fiz entre a classe TNota e TItensNota?

Tentando solucionar essas dificuldades muitos criam soluções engenhosas, mas pouco flexíveis. Neste artigo quero introduzir o Framework tiOPF (TechInsight Object Persistence Framework), que é responsável pela persistência de objetos. Além disso, ele oferece também meios de se criar interfaces de usuário de forma bem produtiva.

Vamos construir um pequeno exemplo que permita a manutenção de uma lista de contatos, contendo um nome, e-mail e uma lista de telefones. Veremos como será feito o mapeamento objeto relacional e como o tiOPF lida com o relacionamento entre um contato e seus possíveis telefones.

O framework tiOPF

O tiOPF é um framework que realiza o mapeamento objeto relacional para nós, assumindo a responsabilidade de persistir objetos de negócio em tabelas de um banco de dados. Quando desenvolvemos um sistema aplicando a orientação a objetos nos deparamos com um problema chamado impendância objeto/relacional (que comentei anteriormente). Resumidamente esse problema representa a diferença existente entre esses dois mundos.

Resumo DevMan

De que trata o artigo:

Mostrar como utilizar o Framework tiOPF para realização de mapeamento objeto relacional e como utilizá-lo no desenvolvimento de um sistema orientado a objetos, ilustrando como persistir esses objetos de negócio.

Para que serve:

O tiOPF é um Framework que realiza a persistência de objetos, oferecendo assim meios para inserir, recuperar, localizar e excluir de um banco de dados (seja ele relacional ou arquivo texto) objetos de negócio implementados em Delphi.

Em que situação o tema é útil:

No desenvolvimento orientado a objetos em Delphi, é muito comum que os desenvolvedores criem suas próprias soluções para esse requisito. Mas na grande maioria das vezes essas não são reutilizáveis e nem apresentam recursos suficientes. O tiOPF é um Framework pronto para essas situações.

Mapeamento Objeto / Relacional em Win32:

Com certeza as maiores dificuldades enfrentadas por um desenvolvedor Delphi que escolhe aplicar a orientação a objetos em seu sistema estão concentradas nas camadas de persistência e apresentação, fazendo com que em muitos dos casos, um desânimo cresça e o interesse pela orientação a objetos desapareça, o que leva a sistemas Delphi não terem a qualidade de plataformas como .NET e Java, que pregam a OO em praticamente qualquer tipo de aplicação. Esse é um mito que vamos derrubar hoje. Conheça aqui o tiOPF e veja como seus recursos podem agilizar e facilitar o uso de objetos em suas regras de negócio. Vamos desenvolver um exemplo prático que irá expor os conceitos básicos que permitirão o uso imediato da ferramenta no seu dia a dia.

Para exemplificar a situação, imagine que temos duas classes, TPessoa e TEndereco. Elas são relacionadas uma com a outra, ou seja, um objeto TPessoa possui um objeto TEndereco. Agora pense em como essas informações poderiam ser salvas em um banco de dados relacional.

Poderíamos ter uma tabela PESSOA e outra ENDERECO ligadas por uma chave estrangeira e ainda poderíamos ter apenas uma tabela PESSOA que conteria todos os campos necessários. Uma ferramenta ORM cuida de tudo isso para nós, permitindo que mantenhamos nosso foco no que fazemos de melhor, desenvolver realmente o sistema.

O tiOPF é um framework muito completo e bem flexível. Desenvolvido por uma empresa australiana, é muito utilizado naquela região. Altamente maduro e confiável, pois está em produção desde o Delphi 5. Oferece seus próprios controles visuais, que agilizam o desenvolvimento de interfaces com componentes especializados, que tratam desde a validação de um objeto até a visualizações complexas como ListView que atua como um Grid, pronto para manutenção de objetos. Permite que os objetos sejam salvos em diversos bancos relacionais e até mesmo em arquivos XML. Também oferece mecanismos para a criação do banco de dados via código. Com ele

Relacional em Win32

osos recursos de POO, como em .NET e Java

PAULO QUICOLI

Nota do DevMan

A persistência de objetos é ação de poder salvar os dados dos objetos que estão em memória para meios não voláteis, como, por exemplo, um banco de dados relacional, arquivos XML ou até mesmo arquivos texto comuns.

Nota do DevMan

Mapeamento Objeto Relacional: Os dados dos objetos devem ser salvos para um meio físico, que na sua grande maioria é um banco de dados relacional. Um Framework de persistência deve oferecer meios para que seja indicado onde determinada classe será salva, como suas propriedades serão traduzidas, como seus relacionamentos serão tratados. Isso é fazer mapeamento. Geralmente temos uma relatividade entre classe e tabela, onde uma classe será uma tabela e suas propriedades se tornarão campos. Mas nem sempre é assim. Às vezes é possível ter duas classes, por exemplo, Pessoa e Endereço, e a classe a ser transformada em tabela seja apenas Pessoa. Endereço por suas vez se tornaria apenas campos da tabela Pessoa. Isso é chamado de granularidade, que pode ser diferente entre os dois ambientes. Essa diferença é chamada de impedância objeto relacional e, cabe a um bom OPF (framework de mapeamento relacional) diminuir essas diferenças.

também é possível desenvolver aplicativos três camadas, onde objetos irão trafegar entre servidor e cliente. A exigência aqui é que suas classes de negócio herdem de TtiObject, classe base do framework. E claro, você poderá utilizar seus controles Data-Aware mesmo utilizando o tiOPF pois ele disponibiliza um Dataset especial, o TTiDataset que torna os objetos plugáveis nas interfaces DataSet (essa será a técnica usada no artigo). A versão 3 oferece suporte ao Delphi 2010.

Nota

Nota: É chamado de repositório o local onde o código fonte é mantido. Quando o repositório é gerenciado pelo SVN geralmente temos as seguintes pastas em um repositório: Trunk (a pasta com o código fonte final) e Brenches (versões da pasta Trunk, contendo código ainda em desenvolvimento).

Baixando o tiOPF

A versão compatível com o Delphi 2010 ainda não possui um instalador e ainda não teve seu lançamento oficial, que seria a versão 3. Contudo é possível baixar o código fonte direto de seu servidor e então realizar a devida instalação. O código fonte está hospedado em um sistema de controle de versão Subversion (SVN), podendo ser baixado através de um cliente SVN. Devido à simplicidade de uso vamos utilizar o TortoiseSVN para que possamos realizar um Checkout do código fonte mais recente (veja na seção Links onde encontrar o Tortoise). Uma vez baixado o instale. O Tortoise adiciona ao Windows Explorer comandos seus para manutenção de um repositório SVN.

Crie uma pasta chamada tiOPF3 em algum lugar de fácil acesso para você, clique com o botão direito do mouse sobre ela e observe a existência de alguns itens de menu relativos ao Tortoise (**Figura 1**).

Escolha a opção SVN Checkout. Essa opção realiza o download do conteúdo da pasta a ser informada (veja seção Links), como mostra a **Figura 2**.

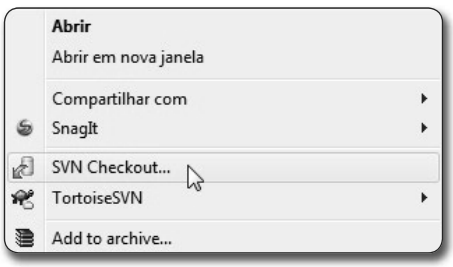


Figura 1. Menus do TortoiseSVN

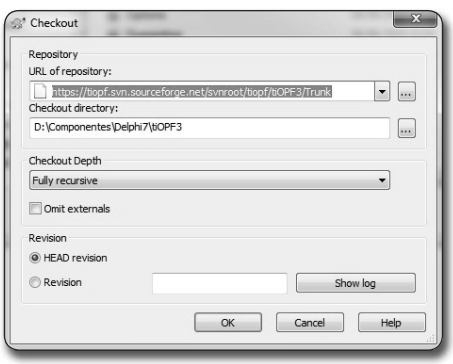


Figura 2. Realizando o Checkout

Instalando o tiOPF

Abra o Delphi 2010 e acesse seu menu File>Open. Na pasta onde foi realizado o Checkout do tiOPF localize a pasta \ Compilers\Delphi2010. Nela, abra o arquivo tiOPF_D2010.groupproj. Ele contém os projetos exibidos na **Figura 3**.

Agora basta clicar com o botão direito, um de cada vez, sobre tiOPFCore2010.bpl, tiOPFOptions2010.bpl, tiOPFGUI2010.bpl e tiOPFGUIDsgn2010.bpl e escolher a opção Compile. Depois disso, clique novamente com o botão direito sobre tiOPFGUI-Dsgn2010.BPL e escolha Install. Agora é necessário adicionar ao Library Path do Delphi alguns caminhos do componente.

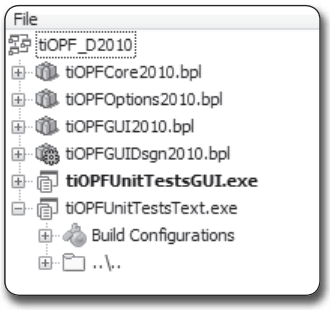


Figura 3. Projetos tiOPF

Para isso clique no menu Tools>Options. Na janela que se abre escolha Delphi Options>Library-Win32. Clique no botão existente ao lado do campo Library Path e inclua as seguintes pastas (assumindo que o componente foi baixado para uma pasta chamada tiOPF3): tiOPF3\Core, tiOPF3\GUI, tiOPF3\Options.

Iniciando o projeto

Inicie um novo projeto no Delphi 2010 (File>New>VCL Forms Application – Win32). Já de início altere o nome do formulário para PrincF e salve tudo da seguinte forma: a unit do formulário como PrincU, o projeto como AgendaDevMedia e o Project Group como DevMedia.

Classes de negócio

Os dados do projeto estarão modelados em classes que representam nossa regra de negócio. Como já foi citado, nosso escopo é o de uma agenda, que possuirá contatos, e estes, telefones. Assim, vamos criar classes representando isso.

Nesse momento adicione uma nova unit ao projeto, clicando com o botão direito no projeto e escolhendo Add New>Unit. Salve com o nome de ModeloClasses. Para aplicarmos o tiOPF é necessário termos duas classes. Uma que representa a entidade em si, e outra que representa uma coleção dessas entidades. Também é necessário que essas classes herdem de tipos específicos do Framework.

Sendo assim, vamos declarar antecipadamente as seguintes classes:

```
type
 TContato = class;
 TContatos = class;

 TTelefone = class;
 TTelefones = class;
```

Nota

Infelizmente ainda não temos para Delphi Win32 um Framework de persistência capaz de utilizar apenas objetos PODO (Plain Old Delphi Objects), que são objetos que simplesmente herdam diretamente de TObject e não necessitam herdar de outra classe para serem persistidos.

Nota

Adicione na seção uses as seguintes units: tiOPFManager, tiAutomap, tiObject, tiOID, tiLog, tiOIDGUID, tiQuery, tiQueryIBX, tiConstants, Windows, Classes, tiFilteredObjectList e SysUtils.

Temos as entidades TContato, TTelefone e suas respectivas coleções. Contudo elas precisam ser complementadas e implementadas. Na **Listagem 1** temos a definição de TContatos.

Observe a **Listagem 1** e veja que nossa coleção herda de TtiFilteredObjectList. Este é um tipo especial de lista do tiOPF que possui recursos de pesquisa. Ou seja, ao utilizá-la será possível fazer uma consulta de contatos, lembrando que esse é um dos objetivos do projeto. Como essa classe é uma coleção de objetos, nada mais lógico do que ela oferecer meios para se adicionar a si própria esses objetos. É exatamente isso que o método Add faz. O método Clear por sua vez marca o estado da coleção em posEmpty, que indica ao Framework que a mesma deve ser esvaziada. Para salvar os objetos da lista temos o método Save, que apenas executa o código herdado e, para carregar todas as informações na lista temos o Read. A coleção também possui a propriedade Items[] para que possamos recuperar um elemento nela contido através de sua posição. Na **Listagem 2** temos a implementação desses métodos.

A **Listagem 2** já teve seu conteúdo explicado, porém quero destacar aqui o método Read. Nele incluímos um registro de Log que marcará quanto tempo cada consulta realizada levou para ser executada, fazemos isso utilizando a função GetTickCount. Esse registro é feito usando o procedimento Log do tiOPF e posteriormente será exibido.

Como já temos a coleção, agora vamos definir a classe TContato. Veja **Listagem 3**.

Na classe TContato, que herda de TtiObject, inicialmente temos as propriedades Nome, Email e Telefones juntamente com seus métodos de acesso. Vejam que Telefones é do tipo TTelefones, uma coleção. Sendo assim, um Contato poderá possuir mais de um telefone, e os mesmos serão trazidos assim que localizarmos um contato. A propriedade Owner indica qual é a lista de objetos a que ele pertence. Os demais métodos: GetOwner, SetOwner, Create, Destroy, IsValid, SetObjectState e AssignClassProps serão vistos na **Listagem 4.**

Listagem 1. Coleção de contatos

```
TContatos = class(TtiFilteredObjectList)
private
  function GetItems(i: integer): TContato;
  procedure SetItems(i: integer;
   const Value: TContato);
public
  property   Items[i:integer]: TContato read
  GetItems write SetItems;
  procedure  Add(AObject: TContato);
   reintroduce;
  procedure  Clear; override;
  procedure  Read; override;
  procedure  Save; override;
end
```

Listagem 2. Implementação de TContatos

```
procedure TContatos.Add(AObject: TContato);
begin
 inherited Add(AObject);
end;

procedure TContatos.Clear;
begin
 inherited;
 ObjectState:= posEmpty;
end;

function TContatos.GetItems(i: integer): TContato;
begin
 result:= TContato(inherited GetItems(i));
end;

procedure TContatos.SetItems(i: integer;
 const Value: TContato);
begin
 inherited SetItems(i, Value);
end;

procedure TContatos.Read;
var
 INow: DWord;
begin
 INow:= GetTickCount;
 inherited;
 Log('Tempo para carregar os contatos: ' +
   IntToStr(GetTickCount - INow));
end;

procedure TContatos.Save;
begin
 inherited;
end;
```

Nota do DevMan

GetTickCount é uma função da API do Windows que retorna o tempo (em milissegundos) decorrido desde que o sistema foi iniciado. É muito utilizado para medições de performance, sendo executado antes de um processamento a ser verificado e posteriormente à execução do mesmo, subtraindo-se então os dois valores obtidos para gerar o tempo levado.

O método AssignClassProperties é necessário quando temos em nossa classe alguma(s) propriedade(s) que seja(m) de tipos definidos como classes de negócio. Por exemplo, TContato possui TTelefones. Esse método é utilizado para que, corretamente, sejam criados ou referenciados os elementos da lista no novo objeto a ser copiado ou clonado, ou seja, esse método é utilizado internamente quando desejamos clonar um objeto. Já no construtor da classe instanciamos a lista de telefones e já definimos seu proprietário (Owner). No Destroy, para liberar recursos e prevenir Memory Leaks, retiramos a lista de telefones da memória. O método isValid é muito interessante. Ele é definido na classe base e é utilizado para que possamos realizar alguma validação antes de persistir um objeto. No caso estamos verificando se o Nome é vazio, se for, adicionamos um erro à sua lista interna informando à propriedade que a originou o erro e uma mensagem indicativa para o usuário. Sendo assim, um objeto só é válido se sua lista de erros estiver vazia. Já o método SetObjectState está aí por que a versão atual para o Delphi 2010 ainda não está completa. O Framework é estável e funcional, mas algumas coisas consideradas como de acabamento ainda precisam ser feitas. Esse método é utilizado internamente pelo Framework para ajustar o estado da lista de telefones que o contato possui. O tiOPF mantém o estado dos objetos, por exemplo, para saber quando alterar, excluir ou incluir um objeto novo.

Continuando com os objetos de negócio temos a lista de telefones e o objeto telefone em si. Na **Listagem 5** temos TTelefones.

Se compararmos a lista de telefones com a lista de contatos veremos semelhanças e diferenças. TTelefones herda de TtiObjectList, que é outra classe do Framework que representa uma coleção de objetos. Mas é uma classe mais simples, que não oferece mecanismo de pesquisa. A utilizamos porque não faremos busca alguma

Listagem 3. Classe TContato

```
TContato = class(TtiObject)
private
  FEmail: string;
  FOwner: TContatos;
  FTelefones: TTelefones;
  FNome: string;
  procedure SetEmail(const Value: string);
  procedure SetNome(const Value: string);
  procedure SetOwner(const Value: TContatos); reintroduce;
  function GetOwner: TContatos; reintroduce;
  procedure SetTelefones(const Value: TTelefones);
  procedure SetObjectState(const AValue: TPerObjectState); override;
public
  constructor Create; override;
  destructor Destroy; override;
  property Owner : TContatos read FOwner write SetOwner;
  function IsValid(const AErrors: TtiObjectErrors): boolean; override;
  procedure   AssignClassProps(pSource: TtiObject); override;
published
  property Nome: string read FNome write SetNome;
  property Email: string read FEmail write SetEmail;
  property Telefones: TTelefones read FTelefones write SetTelefones;
end;
```

Listagem 4. Implementação de TContato

```
procedure TContato.AssignClassProps(pSource: TtiObject);
begin
 Telefones.Assign(TContato(pSource).Telefones);
end;

constructor TContato.Create;
begin
 inherited;
 Telefones := TTelefones.Create;
 Telefones.Owner := self;
end;

destructor TContato.Destroy;
begin
 Telefones.Free;
 inherited;
end;

function TContato.IsValid(const AErrors: TtiObjectErrors): boolean;
begin
 if Nome = '' then
  AErrors.AddError('Nome', 'Você deve informar o nome do contato.');
 result:= AErrors.Count = 0;
end;

procedure TContato.SetObjectState(const AValue: TPerObjectState);
begin
 inherited;
 Assert(Telefones.TestValid, CTIErrorInvalidObject);
 if (AValue = posDeleted) and (Telefones.ObjectState = posDelete) then
  Telefones.ObjectState:= posDeleted;
end;

function TContato.GetOwner: TContatos;
begin
 result:= TContatos(inherited GetOwner);
end;

procedure TContato.SetOwner(const Value: TContatos);
begin
 FOwner := Value;
end;
```

por telefone. Não temos os métodos Clear, Read e Save porque quem irá controlar quando recuperar ou salvar telefones será seu Owner, no caso um objeto do tipo TContato. Dessa forma, quando salvarmos um contato, automaticamente sua lista de telefones será salva/atualizada no banco de dados. Temos de diferente o método GetOID. Esse método irá retornar o identificador do objeto a quem essa lista pertence. Na **Listagem 6** temos a implementação desses métodos.

Como pôde ser visto na **Listagem 6**, nada tão complexo e sua funcionalidade já foi explicada. Vamos agora para a classe TTelefone, vista na **Listagem 7.**

Semelhantemente à classe TContato, TTelefone herda de TtiObject. Nela também temos a propriedade Owner indicando qual a lista que possui o objeto, temos também a propriedade Numero que irá armazenar o número do telefone em si, e por fim a função isValid que é utilizada para validar o telefone. Na **Listagem 8** vemos as demais implementações.

A função isValid de TTelefone tem a responsabilidade de verificar duas situações: se o telefone está vazio e se possui caracteres não permitidos. Caso algum dos casos seja verdade, adicionamos o erro com sua mensagem adequada e o objeto se torna inválido. Observe a forma como estamos validando os caracteres da propriedade número. Através de um laço for in percorrer cada caractere do número, pesquisando-o numa lista de caracteres válidos. Se o caractere não for encontrado nessa lista, o número é inválido. Essa lista é representada através da variável caracteresValidos do tipo TStringList, que deve ser declarada no escopo global do formulário e que é instanciada na seção initialization da unit e destruída na seção finalization, ambas seções são localizadas antes do end final. Essas duas seções são executadas respectivamente quando a unit é carregada e descarregada da memória. Veja na **Listagem 9** sua utilização.

Nota do DevMan

OID, ou Object Identification, é um termo comum utilizado para representar a propriedade que torna o objeto único em sua coleção ou entre todos os objetos do sistema, é utilizado para garantir a unicidade do mesmo. Por possuir semântica similar ao conceito das chaves primárias de um banco de dados relacional, no trabalho de mapeamento esse identificador é geralmente mapeado para a chave primária da tabela que representa a classe.

Criando o mapeamento

O mapeamento basicamente é a tarefa de dizer ao Framework em que tabela

Listagem 5. Definição da coleção de telefones

```
TTelefones = class(TtiObjectList)
 private
  FOwner: TContato;
  function GetItems(i: integer): TTelefone;
  procedure SetItems(i: integer;
   const Value: TTelefone);
  procedure SetOwner(const Value: TContato);
 protected
  function   GetOID: TtiOID; override;
public
  property Items[i:integer]: TTelefone read
   GetItems write SetItems;
  procedure Add(AObject: TTelefone);
  reintroduce;
  property Owner: TContato read FOwner write
   SetOwner;
 published
 end;
```

Listagem 6. Implementação dos métodos de TTelefones

```
procedure TTelefones.Add(AObject: TTelefone);
begin
 inherited Add(AObject);
end;

function TTelefones.GetItems(i: integer): TTelefone;
begin
 result:= TTelefone(inherited GetItems(i));
end;

function TTelefones.GetOID: TtiOID;
begin
 result:= Owner.OID;
end;

procedure TTelefones.SetItems
 (i: integer; const Value: TTelefone);
begin
  inherited SetItems(i, Value);
end;

procedure TTelefones.SetOwner
 (const Value: TContato);
begin
 FOwner := Value;
end;
```

Listagem 7. Classe TTelefone

```
TTelefone = class(TtiObject)
 private
  FOwner: TTelefones;
  FNumero: string;
  procedure SetNumero(const Value: string);
  procedure SetOwner(const Value: TTelefones);
   reintroduce;
  function GetOwner: TTelefones; reintroduce;
 published
 public
  property Owner: TTelefones read GetOwner
    write SetOwner;
  function IsValid(const AErrors: TtiObjectErrors):
    boolean; override;
 published
  property Numero: string read FNumero write
   SetNumero;
end;
```

Listagem 8. Implementação da validação de TTelefone

```
function TTelefone.GetOwner: TTelefones;
begin
  result:= TTelefones(inherited GetOwner);
end;

function TTelefone.IsValid(const AErrors:
   TtiObjectErrors): boolean;
var
 s: string;
begin
  inherited IsValid(AErrors);

  if Numero = '' then
    AErrors.AddError('Numero', 'Você deve informar
       um número para o telefone');

for s in Numero do
  begin
    if caracteresValidos.IndexOf(s)  < 0 then
    begin
      AErrors.AddError('Numero', 'O número do
         telefone contém caracteres inválidos');
      break;
    end;
  end;
  result:= AErrors.Count = 0;
end;
```

e campos uma determinada classe será persistida. O tiOPF possui uma classe que atua como um gerenciador de objetos. Esse gerenciador é representado pela instância GTIOPFManager e é nela que podemos realizar o mapeamento. Para este exemplo vamos utilizar um recurso chamado AutoMapping, que é um mapeamento bem básico das classes e propriedades. Ele é indicado para sistemas pequenos que não possuem grande volume de dados, visto que ao utilizá-lo, não especificamos como é que devem ser geradas as sentenças SQL para obtenção de coleções e coisas do tipo. Vamos criar então um procedimento global que irá definir esse mapeamento. Veja **Listagem 10**.

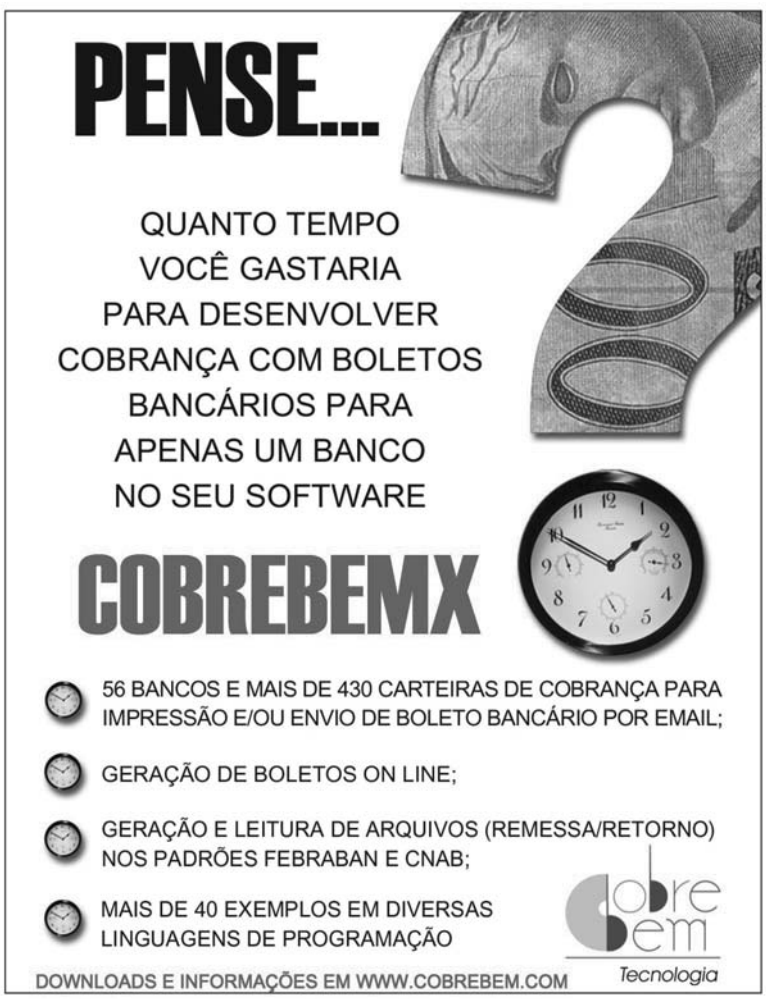
Nota

O tiOPF oferece outros meios de mapeamento que permitem ao desenvolvedor otimizar ao máximo o SQL gerado pelo Framework. Esses meios são chamados de Visitors e HardCode. Em Visitors o desenvolvedor aplica o padrão de projeto Visitor (veja seção Links) para controlar a hierarquia de objetos que pode ser trazida. No HardCode o desenvolvedor tem a liberdade de expressar manualmente as sentenças SQL que serão geradas.

A classe GTIOPFManager contém o objeto ClassDBMappingMgr que deve conter a especificação de um AutoMapping. Observe que utilizamos o método RegisterMapping para isso. Nele especificamos a classe a ser mapeada, o nome da tabela, a propriedade, o nome do campo e um parâmetro opcional que indica que tipo de chave o campo representa. Vejamos o mapeamento de TContato. A primeira propriedade é OID, mas nós não criamos essa propriedade em

Listagem 9. Lista caracteresValidos

```
initialization
  caracteresValidos := TStringList.Create;
  caracteresValidos.Add('0');
  caracteresValidos.Add('1');
  caracteresValidos.Add('2');
  caracteresValidos.Add('3');
  caracteresValidos.Add('4');
  caracteresValidos.Add('5');
  caracteresValidos.Add('6');
  caracteresValidos.Add('7');
  caracteresValidos.Add('8');
  caracteresValidos.Add('9');
  caracteresValidos.Add('');
  caracteresValidos.Add('(');
  caracteresValidos.Add(')');
  caracteresValidos.Add('-');
finalization
  caracteresValidos.Free;
```

Listagem 10. Mapeamento das classes

```
procedure RegisterAutoMappings;
begin
 with GTIOPFManager.ClassDBMappingMgr do
 begin
   RegisterMapping(TContato,'CONTATO','OID', 'ID_CONTATO', [pktDB] );
   RegisterMapping(TContato,'CONTATO','Nome','NOME');
   RegisterMapping(TContato,'CONTATO','Email','EMAIL');
   RegisterCollection(TContatos, TContato);
   RegisterMapping(TTelefone,'TELEFONE','OID', 'ID_TELEFONE', [pktDB] );
   RegisterMapping(TTelefone,'TELEFONE','Owner.OID','ID_CONTATO',[pktFK]);
   RegisterMapping(TTelefone,'TELEFONE','Numero','NUMERO');
   RegisterCollection(TTelefones, TTelefone);
 end;
end;
```

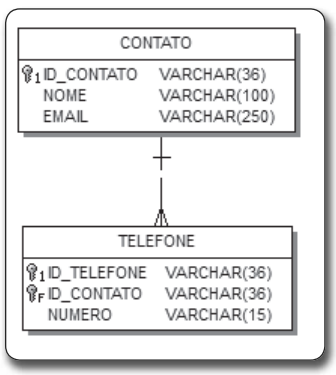


Figura 4. Tabelas do banco de dados

TContato. Ela está definida em TtiObject e representa a identificação do objeto, a chave primária. Ela será salva na tabela CONTATO com o nome de ID_CONTATO. Esse campo é a chave primária, portanto o marcamos com [pktDB]. As outras propriedades de TContato são mapeadas de forma similar. Na sequência informamos ao Framework que TContato será mantido por uma lista chamada TContatos. O mapeamento de TTelefone é um pouco diferente. Veja que temos agora a propriedade Owner.OID sendo mapeada para o campo ID_CONTATO da tabela TELEFONE. Ele é nossa chave estrangeira com a tabela CONTATO. Semelhantemente também registramos a coleção de telefones. Com essas informações podemos já criar o banco de dados. Vou utilizar aqui o Firebird. Crie um banco chamado BANCO.FDB e nele crie a estrutura mostrada na **Figura 4.**

Com o banco de dados criados já podemos realizar uma conexão. Acesse o formulário principal e em sua seção uses adicione as seguintes units: tiOPFManager, tiDialogs, tiUtils, e tiLog. Agora no evento OnCreate do formulário adicione o código da **Listagem 11**.

Ao criar o formulário nós estamos definindo qual camada de persistência será utilizada, isto é, como será feito o acesso ao banco de dados. Para este exemplo passamos a sigla IBX, identificando que vamos utilizar o pacote IBX e consequentemente o banco será Firebird ou Interbase. Mas o tiOPF oferece vários meios de acesso, temos as seguintes opções:

- Interbase/Firebird via IBX ;
- Firebird via FBLib;
- Firebird via ZeosLib;
- Oracle via DOA;
- MS Access e MS SQL-Server via ADO;
- Paradox via BDE;
- XML via MSDOM ou DOM do Free Pascal;
- Arquivos CSV.

Depois informamos os parâmetros para conexão do banco, através do método ConnectDatabase que já realiza a conexão, para então chamar o método RegisterAutoMappings que registrará o mapeamento.

Criando a interface

Todo desenvolvedor Delphi sabe da praticidade que os controles Data-Aware proporcionam. Eles se ligam automaticamente aos DataSets, retirando de nós o trabalho de exibição dos dados. Mas no momento estamos desenvolvimento um aplicativo que possui objetos "puros", e não descendentes de TDataSet. Pensando nisso a equipe que desenvolveu o tiOPF incluiu também dois tipos especiais de TDataset. O TtiDataSet e o TtiNestedDataSet. Ambos fazem uma ponte entre uma coleção de objetos e os controles Data-Aware, permitindo ligar e manipular os objetos como se fossem registros de um TDataset. Vamos utilizá-los.

Adicione ao projeto um novo Data Module e o salve com o nome de dtm. Nele adicione um controle TtiDataSet e um TtiNestedDataset, ambos estão localizados

Listagem 11. Conectando ao banco de dados

```
procedure TPrincF.FormCreate(Sender: TObject);
begin
 GTIOPFManager.DefaultPersistenceLayer
 Name:='IBX';
 GTIOPFManager.ConnectDatabase('localhost:
  D:\Projetos\AgendaDevMedia\Banco\banco.fdb',
  'SYSDBA','masterkey');
 RegisterAutoMappings;
end;
```

na seção TechInsite Dataset da Tool Palette. Nomeie cada um respectivamente como tiClientes e tiTelefones. Adicione também dois TDatasource de nomes dsClientes e dsTelefones. Ligue cada um a seus respectivos DataSets.

Para que ambos funcionem é necessário adicionar a eles campos com os mesmos nomes e tipos existentes na classe que representarão. Sendo assim dê um clique duplo sobre tiClientes. Seu Fields Editor irá aparecer. Nele crie os seguintes campos: Nome, string, tamanho 100; Email, string, tamanho 150; Telefones, DataSet. Observe que até mesmo um campo Telefones é criado para representar a lista de telefones existente, o detalhe é que esse campo é do tipo TDataset. Vamos utilizar agora o tiTelefones para manter a lista de telefones de TContato. Veja que esse Dataset é do tipo TtiNestedDataset, ou seja, é um Dataset que estará aninhado em tiClientes. Nele adicione o seguinte campo: Numero, string, tamanho 15. Também é necessário ligar sua propriedade DatasetField ao campo Telefones de tiClientes, que aparecerá automaticamente no Object Inspector. Feito isso, acesse o menu Project>Options>Forms e faça com que o Data Module seja o primeiro da lista, assim ele será criado antes do formulário principal. Agora acesse o formulário principal, inclua

uma referência ao Data Module e o deixe como mostra a **Figura 5**.

Figura 5. Formulário principal

Como temos suporte aos controles DataAware, estamos utilizando dois DBGrids mesmo. Como vamos lidar com vários contatos, a primeira coisa a se fazer é adicionar ao formulário um campo privado do tipo TContatos:

```
FListaContatos: TContatos;
```

Vamos também adicionar um método privado que irá carregar a lista de objetos disponíveis para nós. Veja **Listagem 12.**

O procedimento AbrirContatos limpa a lista de Contatos através do método Clear e na sequência executa o método Read que carrega do banco de dados os registros salvos. Então verificamos se tiClientes já está ativo, se estiver mandamos que seja atualizado. Agora é necessário utilizar esse método no construtor do formulário. Assim, vamos alterá-lo (**Listagem 13**).

Como já foi visto, o código da **Listagem 13** mostra como fazemos a conexão com o banco de dados. Agora também instanciamos a lista de contados (FListaContatos) e fazemos a ligação entre o TtiDataSet e a lista em si. Isso é feito através do método LinkObject. Para o TtiNestedDataset apenas informamos o tipo de classe que ele irá conter, no caso, TTelefone. E então chamamos o método AbrirContatos. Agora vamos fazer o formulário de edição de um contato. Adicione um novo formulário ao projeto, lhe dando o nome de EdicaoContatosF. Configure-o como mostra a **Figura 6.**

Adicione uma referência ao Data Module e ligue o DBGrid e os DBEdits aos respectivos Datasources. Nesse formulário também temos um TMemo que será utilizado para exibir as mensagens de validação. No ActionList crie uma ação de nome acConfirmar. Nela insira o código a seguir (ligue essa ação ao botão Confirmar):

```
ModalResult := mrOk;
```

Agora efetue um clique duplo no botão cancelar e nele inclua o seguinte código:

```
ModalResult := mrCancel;
```

Assim, clicando em ambos os botões o formulário será fechado, apenas indicando através do ModalResult se a operação foi confirmada ou não. Agora vamos implementar algo prático nesse formulário. Vamos habilitar o botão confirmar apenas quando o objeto for válido. Assim, no evento OnUpdate do ActionList1 adicione o seguinte código:

```
acConfirmar.Enabled := dtm.tiClientes.ObjectList.Dirty and
 dtm.tiClientes.GetActiveItem.IsValid(memoErros.Lines);
```

Nele estamos habilitando a ação de confirmar apenas se a lista de objetos tiver sofrido alteração (estado Dirty) e se o item ativo de tiClientes for válido. Essa verificação é feita através do método IsValid que pode receber como parâmetro um objeto do tipo TStrings. Assim, caso tenha erros, eles serão adicionados nesse TStrings, que em nosso caso é o conteúdo do TMemo. Agora que temos um formulário de edição podemos continuar o trabalho no formulário principal.

Incluindo, editando e excluindo objetos

Volte ao formulário principal e dê um clique duplo sobre o botão Incluir. Em seu evento OnClick adicione o código da **Listagem 14.**

O código é bastante simples. Veja que trabalhamos nele como se não existisse objeto algum. Damos um Insert em tiClientes e executamos o formulário de edição, que também está conectado a tiClientes. Se o usuário confirmou a inclusão, efetuamos um Post e solicitamos à lista de contatos que salve o que foi incluído. Vejamos agora a edição de um contato. No botão Editar adicione o código da **Listagem 15**.

Figura 6. Edição de Contato

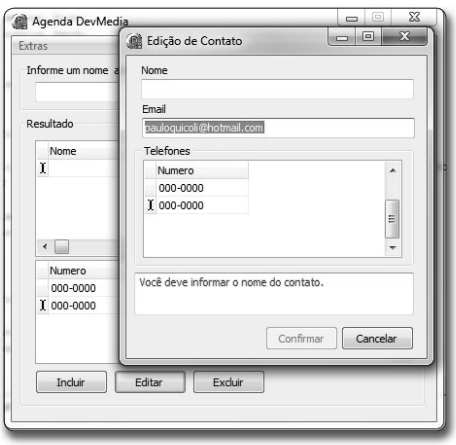


Figura 7. Formulário principal em ação

Listagem 12. AbrirContatos

```
procedure TPrincF.AbrirContatos;
begin
 FListaContatos.Clear ;
 FListaContatos.Read ;
 if dtm.tiClientes.Active then
  dtm.tiClientes.Refresh
 else
  dtm.tiClientes.Open;
end;
```

Listagem 13. Carregando os contatos

```
procedure TPrincF.FormCreate(Sender: TObject);
begin
 GTIOPFManager.DefaultPersistenceLayerName:='IBX';
 GTIOPFManager.ConnectDatabase(
      'localhost:D:\Projetos\AgendaDevMedia\
      Banco\banco.fdb',
      'SYSDBA','masterkey');
 RegisterAutoMappings;

 FListaContatos := TContatos.Create;
 dtm.tiClientes.LinkObject(FListaContatos,TContato);
 dtm.tiTelefones.ObjectClass := TTelefone;
 AbrirContatos;
end;
```

Veja que a única diferença em relação ao inserir é que agora damos um Edit em tiClientes. A exclusão também é bem simples. Adicione o código da **Listagem 16** no evento OnClick do botão Excluir.

Agora nós solicitamos uma confirmação, e esta sendo confirmada, executamos o Delete de tiClientes e pedimos que a lista seja salva. Execute a aplicação, insira, exclua e altere alguns objetos. Você irá notar que a lista de telefones é trazida automaticamente. Não é necessário de nossa parte nenhum código para manter o relacionamento entre os dois. Veja na **Figura 7** o programa em funcionamento.

Como pode ser visto até mesmo a validação já funciona. Vamos continuar com a pesquisa de contatos.

Pesquisando contatos

A procura também é muito simples. Você se lembra que nossa lista de contatos é do tipo TtiFilteredObjectList. Muito bem, esse tipo disponibiliza alguns métodos de pesquisa, denominados Criteria. Esses Criterias são transformados em sentenças SQL pelo Framework, por isso possuem nomes bem relacionados a isso. Dê um clique duplo sobre o botão Localizar e nele adicione o seguinte código:

```
procedure TPrincF.Button1Click(Sender: TObject);
begin
 FListaContatos.Criteria.ClearAll;
 FListaContatos.Criteria.AddLike('Nome', EditBusca.
  Text + '%');
 AbrirContatos;
end;
```

Veja novamente a simplicidade de código. A lista de contatos possui uma pro priedade que é uma coleção de Criterias. Ao fazer uma pesquisa nós limpamos essa lista e adicionamos um Criteria do tipo Like. Nele passamos como parâmetro a propriedade a ser localizada e o texto a ser procurado. Execute a aplicação e veja como tudo funciona perfeitamente.

Funções auxiliares

O tiOPF oferece alguns métodos que podem vir a auxiliar um possível debug da aplicação. O primeiro que quero demonstrar é a visualização interna de uma lista de objetos. No formulário principal nós adicionamos um TMainMenu. Nele adicione um MenuItem Extras e dentro deste, um outro chamado Inspecionar Objetos. No clique deste último MenuItem adicione o seguinte código:

```
tiShowString(FListaContatos.AsDebugString);
```

Este irá exibir em forma de debug informação sobre a lista de objetos passada como parâmetro, bem como informações de cada elemento que ela possuir. Veja o resultado na **Figura 8**.

Performance é preocupação de todos. E para auxiliar o desenvolvedor, o Framework também oferece a visualização de log. Nesse log é possível detectar a SQL que é gerada, bem como as mensagens geradas pelo componente de conexão utilizado. Assim, é possível acompanhar tudo o que acontece até seu objeto chegar ao banco de dados. Para ativarmos esse log é necessário adicionarmos à unit do projeto uma referência à tiLogToGUI. Para isso acesse o menu Project>View Source e logo abaixo à declaração de Forms em uses, inclua tiLogToGUI. Mas isso não é só. A exibição desse log é opcional e é feita por parâmetros. No caso, para ativar essa exibição é preciso passar o parâmetro –lv para a linha de comando do projeto. Faça

Listagem 14. Criando um novo contato

```
procedure TPrincF.btnIncluirClick(Sender: TObject);
begin
 dtm.tiClientes.Insert;
 if EdicaoContatosF.ShowModal = mrOk then
 begin
  dtm.tiClientes.Post;
  FListaContatos.Save;
 end
 else
  dtm.tiClientes.Cancel;
end;
```

Listagem 15. Editando um contato

```
procedure TPrincF.btnEditarClick(Sender: TObject);
begin
 dtm.tiClientes.Edit;
 if EdicaoContatosF.ShowModal = mrOk then
 begin
  dtm.tiClientes.Post;
  FListaContatos.Save;
 end
 else
  dtm.tiClientes.Cancel;
end;
```

Listagem 16. Excluindo um contato

```
procedure TPrincF.btnExcluirClick(Sender: TObject);
begin
 if Application.MessageBox('Confirma a exclusão
   do Contato?','Atenção',
    MB_YESNO + MB_ICONQUESTION
     + MB_DEFBUTTON2) =  ID_YES then
 begin
  dtm.tiClientes.Delete;
  FListaContatos.Save;
 end;
end;
```

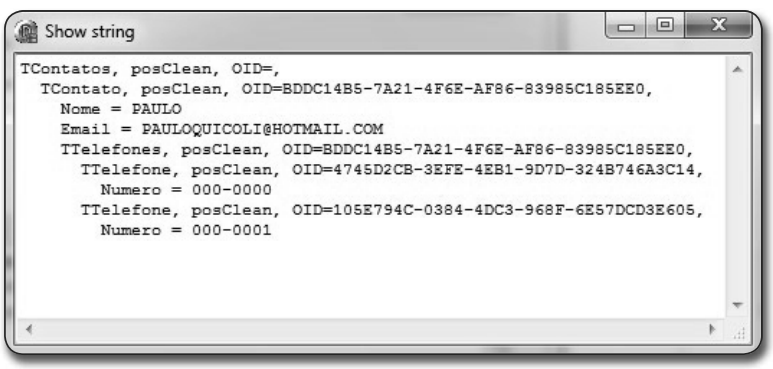


Figura 8. Inspecionando objetos

isso acessando o menu Run>Parameters. No campo Parameters adicione –lv e execute a aplicação. Ao executar agora será exibida uma janela com várias opções de visualização. Se desejar ver o SQL gerado, marque SQL. Veja a **Figura 9**.

Como mostra a **Figura 9** é possível ver a mensagem de log que colocamos no método Read da lista de contatos.

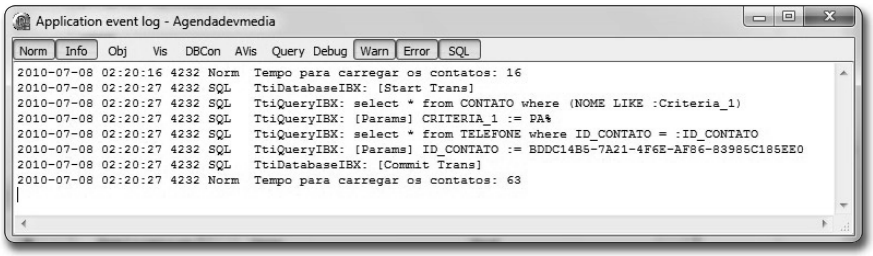


Figura 9. Exibição do Log

Conclusão

O Framework tiOPF possui vários recursos adicionais que não puderam ser exibidos aqui, como, por exemplo, a possibilidade de ser utilizado em um ambiente três camadas e até mesmo Web.

Agora não temos mais desculpas para não aplicar a orientação a objetos em nossos sistemas. Podemos gerar relatórios, continuar utilizando aquele componente de terceiro que temos, isto porque o tiOPF oferece suporte ao TDataSet. Espero que este artigo tenha despertado o interesse sobre o assunto. Até mais.

Paulo Quicoli
pauloquicoli@gmail.com
Editor técnico das revistas ClubeDelphi, .Net Magazine e WebMobile Magazine. Formado em processamento de dados pela FATEC-TQ. Atua como analista e desenvolvedor da ControlM Informática (www.controlm.com.br) e professor de linguagem de programação do curso técnico em processamento de dados do SENAC/Jaboticabal. Utiliza Delphi desde sua primeira versão, entusiasta do desenvolvimento orientado a objetos, tem publicado vários artigos sobre o assunto. Blog: www.pauloquicoli.spaces.live.com .

Tortoise
http://tortoisesvn.net/downloads

Site tiOPF
http://tiopf.sourceforge.net/

Repositório SVN tiOPF3
https://tiopf.svn.sourceforge.net/svnroot/tiopf/tiOPF3/Trunk

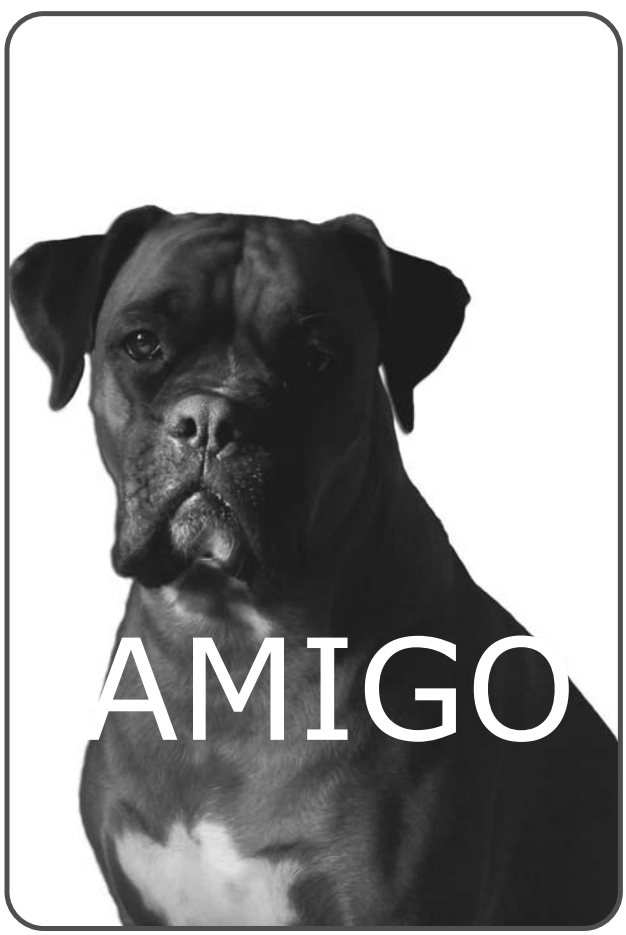
Padrão Visitor
http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/pat/visitor.htm

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback



Existem coisas que não conseguimos ficar sem!

...só pra lembrar, sua assinatura pode estar acabando!

Renove Já!

www.devmedia.com.br/renovacao

Para mais informações:
www.devmedia.com.br/central

DevMedia GROUP

Delphi Prism e ASP.

Reaproveite seus conhecimentos Win32 /

O ASP.NET está nos dando uma excelente mão até agora. Desenvolver para Web nunca foi tão fácil e tão parecido quanto é desenvolver aplicações Desktop. Em especial, quando se fala de passar parâmetros em instruções SQL (tema principal deste artigo), a tarefa fica mais fácil ainda, tão simples que fazemos tudo com cliques de mouse e uso de assistentes.

Nesta etapa da construção do nosso site, vamos modificar a tela principal do site, a Home.aspx. Vamos permitir que o usuário efetue uma busca pelo produto que deseja localizar. De fato, em situações reais nunca saímos exibindo todos os registros do banco de dados, o que seria totalmente ineficaz e sem sentido. Essa busca fará uso de parâmetros no SqlDataSource, que como sabemos, usa efetivamente ADO.NET por baixo (para ser mais exato, uma classe SqlParameter da classe SqlCommand).

Uma das grandes vantagens que vamos tirar proveito aqui na Web, diferente do Desktop, diz respeito ao DataBinding. Como todo o processamento no servidor ocorre uma única vez, no Postback, o ASP.NET sabe exatamente quando tem que passar valores para parâmetros SQL. Não é necessário se preocupar em identificar eventos, nem se preocupar com passar os parâmetros manualmente, nem mesmo tratar possíveis ataques de SQL Injection (forma de ataque pelo qual hackers inserem comandos SQL em caixas de entrada), o ASP.NET já pensou em tudo isso para nós.

Além disso, nós veremos como trocar dados entre formulários. No Desktop, por exemplo, temos DataModules que nos permitem acessar um dado em um formulário que pode ser acionado em outro formulário a qualquer momento. O "registro atual" sempre estará lá, selecionado e posicionado. Na Web não existem DataModules, e "atual" é uma palavra extremamente relativa. Se você quer selecionar um produto em uma tela e mostrar todos os seus detalhes em outra, precisa guardar o ID do que foi escolhido para fazer o ASP.NET "lembrar" no outro formulário. A Web é assim.

Falando em vários formulários, já vou deixar bem claro desde agora uma questão típica sobre forms, pois por experiência sei que essa é a primeira pergunta dos desenvolvedores que vêm do ambiente Desktop (além da clássica "Onde está o DataModule"). A questão é: "Como eu faço ou Show ou ShowModal"? Não existe ShowModal na Web, apesar de você conseguir simular algo parecido usando um pouco de JavaScript e AJAX. Mas aqui vai minha dica, o browser não descende de TForm! Não queira fazer

Resumo DevMan

De que trata o artigo:

Lançado pela Embarcadero, o Delphi Prism é a nova forma de desenvolver aplicações para o .NET Framework e Mono utilizando o Object Pascal e o IDE do Visual Studio. Nesta série de artigos, veremos como utilizar o Delphi Prism para criar um site de E-Commerce usando a tecnologia ASP.NET.

Para que serve:

O objetivo principal desta série de artigos é familiarizar os desenvolvedores Delphi Win32 com o ambiente de desenvolvimento do Delphi Prism, o IDE do Visual Studio, bem como introduzir o desenvolvimento de aplicações Web. Dessa forma, faremos vários comparativos entre os dois IDEs e também entre os Frameworks ASP.NET e VCL, servindo como um excelente guia para diminuir a curva de aprendizado ao se aprender ASP.NET baseando-se nos seus prévios conhecimentos em Delphi Win32 (Desktop).

Em que situação o tema é útil:

Esta série será muito útil para quem nunca desenvolveu aplicativo algum para Web, nem mesmo teve um primeiro contato com o Delphi Prism / Visual Studio. Ao desenvolver aplicações para Web, ganhamos inúmeras vantagens se compararmos ao ambiente Desktop: Deploy simplificado, uso de computação nas nuvens, não há necessidade de instalações visto que os usuários acessam um mesmo servidor na internet, não há dependência do sistema operacional, pois usamos um browser, não há preocupação com backups nos clientes, nem altos custos para manutenções e visitas aos clientes etc.

Delphi Prism e ASP. NET:

No primeiro artigo desta série, tivemos uma introdução ao Delphi Prism, seus objetivos e propósitos. A seguir partimos para a criação do banco de dados no SQL Server que será usado ao longo desta série. Na segunda parte iniciamos a criação do Web Site, aproveitando para falar dos fundamentos do ASP.NET e Visual Studio, sempre comparando-os com a VCL e RAD Studio, com o intuito de facilitar o aprendizado. Vimos a estrutura básica de um Web Site e iniciamos a construção do layout básico do site de E-Commerce (chamado "Computer Store"), bem como o sistema de navegação. Na parte anterior, vimos como construir a página principal do site, exibindo a lista de produtos, mostrando como fazer a conexão com o banco de dados. Nesta etapa vamos fazer dois importantes passos do site, tarefas comuns realizadas em qualquer tipo de aplicação Web que trabalhe com banco de dados: pesquisas com parâmetros.

NET

Desktop para criar aplicações Web – Parte 4

Guinther Pauli

na Web as coisas como são no Desktop porque simplesmente a Web não é Desktop e não roda sobre o Windows. Seu usuário pode querer fazer compras de um iPhone, quem sabe. Ou talvez de uma TV 42 polegadas acessando através de um Playstation 3, ou quem sabe da "Google TV". Então não invente, aprenda a limitar-se às regras e imposições da Web para não ter dores de cabeça mais tarde e poder ter certeza que qualquer dispositivo possa acessar seu site. Não faz sentido podermos usufruir de todas as funcionalidades da Web e Cloud Computing se você estampar na primeira página do site que ele só roda em IE 9 no Windows 7, por exemplo. Lembre-se: ASP.NET gera o que chamamos "Full Web", acessível sem a necessidade de instalação de qualquer plugin no cliente (você não verá um ícone de interrogação em seu celular ao acessar sua app no servidor!).

Enfim, trabalhar com múltiplos forms na Web é extremamente diferente do Desktop. Mas não se preocupe, estou aqui par ajudá-lo, e esse é exatamente o objetivo deste mini-curso, facilitar a vida de quem vem do Desktop e precisa saber fazer as mesmas coisas na Web, mas não da forma errada. Fazer diferente, mas aproveitando o que já conhece sobre VCL, Win32, RAD Studio, Firebird etc. Nesse cenário, é bom você também saber (lembrar) que o ASP.NET é state-less, ponto chave da sua performance. Quando você estiver vendo um formulário no browser, não existirá outro formulário em memória no servidor que você possa acessar (ao contrário de aplicações state-full, que na Web simplesmente não funcionam). Na verdade, nem mesmo o formulário que você está vendo existe! De fato, depois de renderizar a página e enviar o HTML ao browser, o ASP.NET destrói tudo até que o usuário faça uma nova requisição.

Em nosso exemplo, nós vamos então redirecionar o usuário para um segundo formulário quando ele encontrar o produto que deseja comprar. Nesse form vamos exibir todos os detalhes do produto escolhido, usando um DetailsView. O DetailsView é um controle introduzido no ASP.NET 2.0 que permite ver um registro na forma de um Grid. É semelhante a um DBGrid da VCL, ou um GridView do ASP.NET, porém invertido: campos da tabela não são as colunas superiores, mas ao lado de cada campo do registro, na vertical.

Vamos então à prática.

Pesquisa por produtos usando parâmetros

Com o Visual Studio aberto, clique em File > Open Web Site e escolha o diretório C:\LOJA, caso este seja o diretório que tenha escolhido para o projeto (o que estamos usando por padrão). No Solution Explorer, dê um duplo clique na página Home.aspx para abrir a página principal do site. No editor, clique na aba Design para exibir a página em modo designer ao invés do código ASPX.

Dentro do ContentPlaceHolder, logo acima do DataList que configuramos na etapa anterior para exibir a lista de produtos, coloque um controle TextBox da aba Standard. Ele será usado para o usuário digitar as iniciais do nome de um produto a ser localizado.

Como este é um mini-curso cuja premissa básica é facilitar o aprendizado na migração para Web, vamos olhar o Desktop. Em Delphi Win32, para uma consulta parametrizada com LIKE para um SELECT como na **Listagem 1**, normalmente faríamos algo como mostra a **Listagem 2**, assumindo que tenhamos dado um FetchParams em um ClientDataSet para via DataSetProvider obter os parâmetros configurados em uma consulta do ADO ou dbExpress.

Listagem 1. Consulta SQL parametrizada com LIKE

```
SELECT [ProductId], [CategoryId], [ProductName],
  [Price]
FROM [Products]
WHERE ProductName
LIKE :ProductName
ORDER BY [ProductName]
```

Listagem 2. Passando parâmetros em Delphi Win32

```
procedure TForm1.btnPesquisarClick(
  Sender: TObject);
begin
  ClientDataSet1.Close();
  ClientDataSet1.Params.ParamByName
    ('ProductName').AsString :=
    Edit1.Text + '%';
  ClientDataSet1.Open();
end;
```

Agora repare como é extremamente simples fazer esse procedimento em ASP.NET. Graças ao mecanismo de DataBinding, não precisamos digitar uma linha de código sequer. Vejamos como. Localize o SqlDataSource responsável por fazer a consulta dos produtos e na sua Smart Tag (a seta que fica no canto superior direito do controle) selecione a opção Configure Data Source. Vá avançando no assistente até chegar na tela que configura a instrução SQL. Clique no botão WHERE, como mostra a **Figura 1**.

Será aberta uma tela como mostra a **Figura 2**. Aqui vamos configurar da seguinte forma as opções:

• Column: Informe a coluna onde vamos estabelecer a condição SQL de WHERE, neste caso, ProductName (vamos pesquisar pelo nome do produto);

• Operator: Escolha LIKE, já que vamos solicitar ao usuário para que digite apenas as letras iniciais do nome do produto a ser localizado;

• Source: Aqui entra em cena uma das grandes maravilhas do ASP.NET. Podemos, via assistente (e não código), informar de onde virá o valor para o parâmetro que estamos especificando (o LIKE ProductName). Veja que a fonte pode ser um controle, o que é semelhante a uma aplicação Desktop, mas também pode ser uma variável de sessão, profile, cookie etc. De fato, é muito comum por exemplo fazermos um SELECT para exibir todos os registros relacionados a um usuário que logou no site, cujo ID estaria armazenado em Sessão, nesse caso o Source seria Session. Para o nosso exemplo, o Source é Control (virá de um controle);

• Control ID: O nome do controle que contém o valor a ser preenchido no parâmetro, em nosso caso escolha TextBox1;

• Default Value: Um valor padrão para o parâmetro, caso exista, em nosso caso vamos manter em branco.

Clique no botão Add e veja que logo abaixo no editor foi adicionado o seguinte item:

```
[ProductName] LIKE '%' + @ProductName + '%'
```

Normalmente em uma tabela com vários registros, onde aplicamos frequentemente uma consulta LIKE em um campo, criamos um índice. Nesse caso, se houver um índice, não seria nada interessante para o SGBDR receber uma instrução como a mostrada anteriormente, configurada por padrão pelo IDE do Visual Studio. Isso porque colocar um "%" antes do parâmetro mata o índice. Clique em Ok e finalize o assistente.

De volta ao designer, selecione o SqlDataSource e abra o editor da sua propriedade SelectQuery. Veja que podemos nesse editor configurar manualmente a instrução SQL e inclusive adicionar alguns outros parâmetros. Retire então o primeiro problemático "%", deixando a instrução como mostrado na **Listagem 3**.

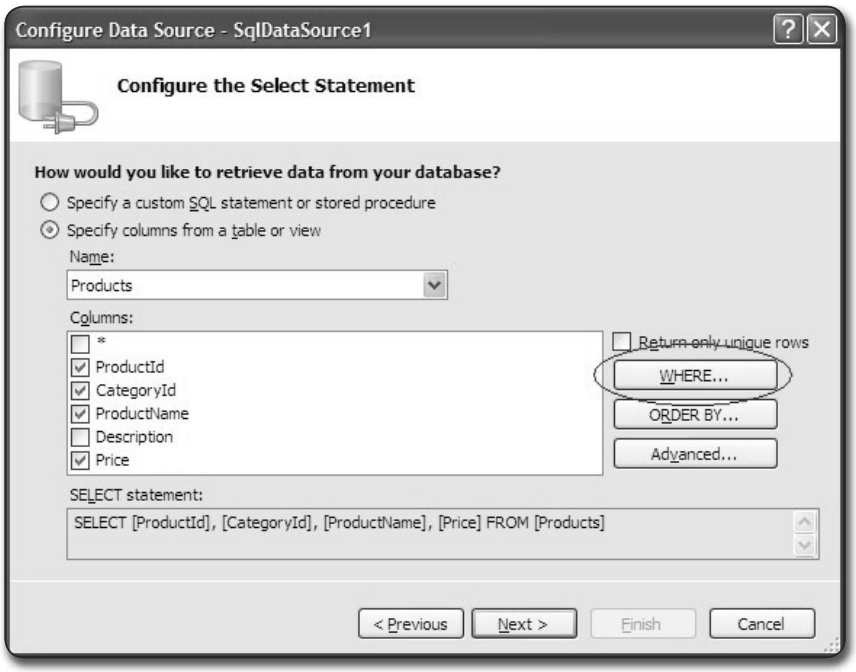


Figura 1. Opção WHERE no assistente

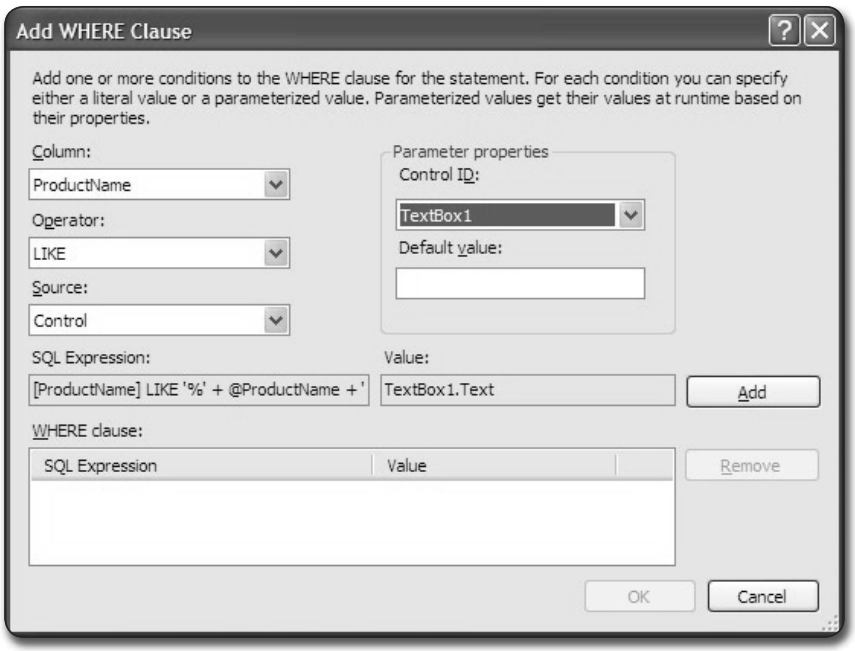


Figura 2. Parametrizando a consulta com um LIKE, usando o assistente

Execute a aplicação, digite no TextBox as letras iniciais do produto que deseja localizar e simplesmente pressione ENTER. Nesse momento, ocorrerá um Postback, ou seja, uma nova requisição ao servidor Web.

Listagem 3. Configurando o Select

```
SELECT [ProductId], [CategoryId], [ProductName],
       [Price]
FROM [Products]
WHERE ([ProductName] LIKE @ProductName + '%')
ORDER BY [ProductName]
```

O ASP.NET fará o DataBinding de toda a página, vendo que preenchemos o valor do parâmetro com o conteúdo do TextBox,

Nota do DevMan

A razão pela qual não podemos usar "%" antes de um parâmetro com LIKE é bastante lógica. Para facilitar, como estamos na sessão EASY, vamos a uma analogia bem simples, mas certamente didática: como você localizaria a palavra "Pádel" em um dicionário (aqueles antigos, de papel)? Abrimos o dicionário no meio e vemos que paramos, digamos, lá pela letra "L". Então sabemos que o que buscamos está na segunda metade da direita do dicionário (já descartamos meio dicionário) e quebramos essa metade novamente. Sobrou 1/4 e paramos mais ou menos pela letra "R". E assim vamos buscando, quebrando em metades, até localizar a palavra. Parece simples, tanto que é exatamente assim que os índices de um SGBDR trabalham, o que chamamos de pesquisa binária (de "bi", dois, daí a analogia com busca por metades). Agora se eu pedisse para você localizar e enumerar no dicionário todas as palavras que contenham a letra "P" em qualquer parte da palavra, o que você teria que fazer? Uma busca sequencial, um a um, certamente, pois precisaria verificar todas as palavras. Traduzindo para o mundo dos bancos de dados relacionais, isso exigiria algo como "Full Table Scan" (pesquisa completa na tabela, de ponta a ponta).

e automaticamente fazendo o vínculo de um com o outro, sem nenhuma linha de código. Veja o resultado na **Figura 3**.

Figura 3. Testando a consulta parametrizada

Mostrando os detalhes do produto

Tanto em aplicações Desktop quanto em aplicações Web, nós nunca trazemos todos os campos de todos os registros do banco de dados para exibir ao usuário em uma tela inicial. Isso até era feito antigamente, mas o BDE dava uma ajuda por baixo dos panos otimizando o tempo de busca e melhorando o feedback com o usuário (pagando o alto custo de prender o cursor do banco de dados).

Na verdade, aplicações Desktop deveriam ser feitas como aplicações Web nesse sentido, e muitas são. O procedimento que normalmente fazemos é realizar uma busca parametrizada, que retornem alguns poucos registros de acordo com a intenção do usuário (foi o que acabamos de fazer). Nessa primeira tela, mostramos os campos essenciais para que o usuário identifique o que quer localizar, então, nada melhor do que mostrar o nome do produto, a foto etc. Mas não todos os campos, isso nós trazemos quando tivermos certeza que o usuário localizou o que deseja comprar. Isso acontece quando ele clica no produto. Se você clicar agora em um produto do nosso site, será redirecionado para a página ShowProduct.aspx (ainda não criada). Veja o formato da URL que o ASP.NET vai gerar para ser acessada pelo browser:

http://localhost:4017/LOJA/ShowProduct.aspx?ProductId=17

Notou o parâmetro? Sim, o ASP.NET passa automaticamente o ID do produto selecionado por parâmetro na URL, na QueryString. Essa é uma das formas de trocarmos informações entre páginas Web em ASP.NET. A troca de dados entre páginas mereceria um capítulo de um livro inteiro, pois é algo que difere totalmente do ambiente Desktop, pois aqui não temos variáveis globais para acessar, nem instâncias de formulários (quem cria e destrói é o ASP.NET). Note que o ID foi passado na URL, sem consumir qualquer recurso do servidor nem guardar nada em memória (lembre-se que na Web cada byte é precioso). Se lembrar, isso aconteceu porque configuramos o HyperLink que exibe o nome do produto com a seguinte expressão, em seu NavigateUrl:

```
Eval("ProductId", "~/ShowProduct.aspx?ProductId={0}")
```

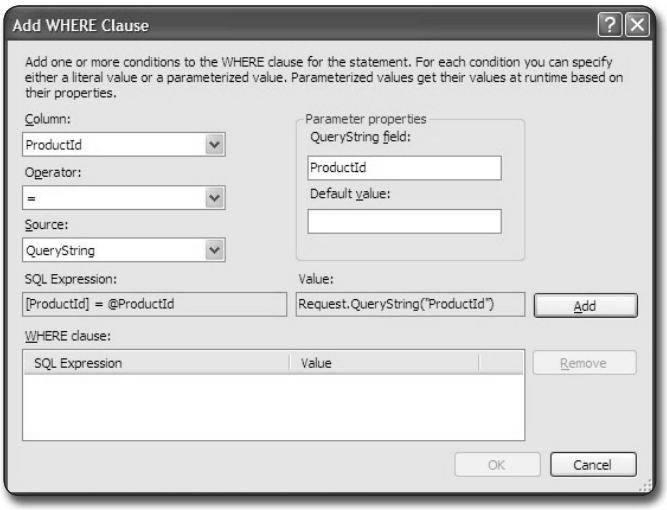


Figura 4. Parâmetro agora vem da QueryString

Vamos então criar a página que irá mostrar os detalhes do produto selecionado. No Solution Explorer, clique de direita sobre o nome do site e escolha Add New Item, escolha Web Form e dê a ele o nome de ShowProduct.aspx. Verifique se a opção para selecionar a Master Page está marcada e na tela seguinte confirme que deseja herdar da Master criada previamente.

Coloque no formulário, dentro do ContentPlaceHolder, um controle DetailsView (guia Data). Na sua Smart Tag, na opção Choose Data Source, escolha New Data Source. No assistente que abrirá escolha Database e vá confirmando todos os passos até chegar na tela do Select. Nesta tela, selecione a tabela Products e clique no * para trazer todos os seus campos (ou mais otimizado, marque todos os campos). Não se preocupe com o fato de estarmos retornando todos os campos, pois é justamente isso que essa tela deve fazer, porém vamos exibir todos os detalhes de um único produto. Então clique na opção WHERE e de forma semelhante ao que fizemos para a pesquisa por nome de produtos configure o parâmetro da consulta como mostrado na **Figura 4**.

Veja que o Source do nosso parâmetro agora não vem de um controle, mas está na URL, ou seja, na QueryString, e se chama ProductId. Pronto, você não precisa fazer mais nada, o primeiro form vai passar o ID do produto para o segundo que vai fazer a consulta parametrizada

Listagem 4. Código ASPX da página de detalhes do produto

```
<%@ Page Language="Oxygene"
    MasterPageFile="~/MasterPage.master"
    AutoEventWireup="true"
    CodeFile="ShowProduct.aspx.pas"
    Inherits="ShowProduct"
    Title="Detalhes do Produto" %>
<asp:Content ID="Content1"
    ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
  <asp:DetailsView ID="DetailsView1"
    runat="server" AutoGenerateRows="False"
    DataKeyNames="ProductId"
    DataSourceID="SqlDataSource1">
    <Fields>
      <asp:BoundField DataField="ProductName" HeaderText="Product"
        SortExpression="ProductName" />
      <asp:ImageField DataImageUrlField="ProductId"
        DataImageUrlFormatString="~/Products/{0}.jpg">
      </asp:ImageField>
      <asp:BoundField DataField="Description" HeaderText="Description"
        SortExpression="Description" />
      <asp:BoundField DataField="Price" HeaderText="Price"
        SortExpression="Price" />
    </Fields>
  </asp:DetailsView>
  <asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%$ ConnectionStrings:LOJAConnectionString %>"
    SelectCommand="SELECT * FROM [Products] WHERE ([ProductId] = @ProductId)">
    <SelectParameters>
      <asp:QueryStringParameter Name="ProductId" QueryStringField="ProductId"
        Type="Int32" />
    </SelectParameters>
  </asp:SqlDataSource>
</asp:Content>
```

exatamente naquele produto usando o ID. Nenhuma linha de código, de novo.

Agora faça alguns ajustes no DetailsView. Primeiro dê uma formatação ao seu gosto usando a opção AutoFormat na Smart Tag do controle. Depois, acesse os Fields do controle, também na Smart Tag, e configure os "Captions" dos Labels que são exibidos ao lado do controle. Para exibir a imagem do produto, simplesmente adicione um campo do tipo ImageField, configure sua propriedade DataImageUrlFormatString para ~/Products/{0}.jpg e DataImageUrlField para ProductId (caso tenha alguma dúvida sobre como devem ficar os campos deste DetailsView, veja a **Figura 5**). O resultado da aplicação já em execução você confere na **Figura 6**. A **Listagem 4** mostra o código final ASPX para esta página (omitindo detalhes de layout não relevantes).

Figura 5. Fields do DetailsView

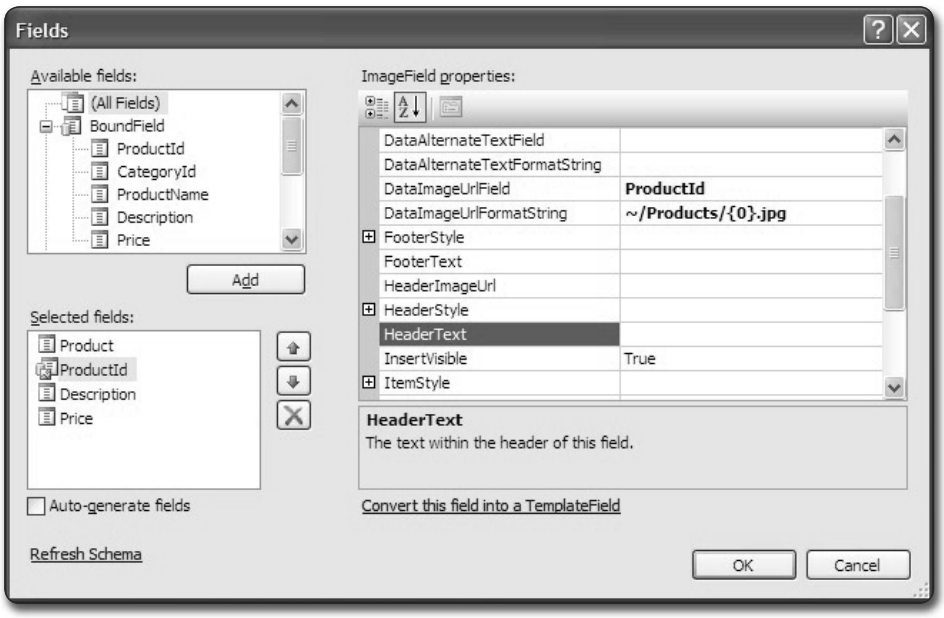


Figura 6. Parâmetro agora vem da QueryString; Web Form mostra os detalhe do produto usando um DetailsView

Nota do DevMan

Há uma diferença em fazer um SELECT * e um SELECT [Nome de Todos os Campos]. No primeiro caso, usando *, o SQL Server terá que fazer uma consulta extra no servidor para em seu catálogo descobrir quais são as colunas atualmente presentes na tabela. Tudo bem, isso leva milésimos de segundo. Mas lembre-se que na Web isso é diferença, ainda mais se tivermos 10 mil usuários acessando o site.

Conclusão

Novamente vou fazer a pergunta que já fiz em conclusões passadas: notou algo interessante até aqui? Nós ainda não digitamos nenhuma linha de código em Delphi Prism! Sim, desenvolver em ASP.NET é tão produtivo e fácil como desenvolver para Desktop, graças aos Server Controls. Nada de se preocupar com HTML, JavaScript, CSS, apesar de esses serem pilares do desenvolvimento Web atual. Não se preocupe, o ASP.NET gera automaticamente a maioria do código cliente para nós, nos padrões mais atuais do mercado. Esta foi uma etapa importante do nosso curso, aprendemos a resolver dois problemas grandes que fazem uma falta tremenda na Web, mas que não poderiam de alguma forma existir, DataModules e formulários modais. Continuamos na próxima edição, continue praticando, grande abraço e até lá!

Guinther Pauli
guinther@devmedia.com.br
guinther.pauli@gmail.com
http://twitter.com/guintherpauli
http://gpauli.com

Desenvolve em Delphi há 15 anos, tendo utilizado a ferramenta desde a versão 1, passando pelo Turbo Pascal e Borland Pascal for Windows. Certificado em Delphi 3, 5, 6, 7, 2005, 2006, Delphi para Web e Delphi para Linux, palestrante em 5 edições da Borland Conference. Certificado Microsoft MCP, MCAD, MCSD.NET, MCTS e MCPD (C#, ASP.NET), utilizando a plataforma .NET desde a versão beta. Editor das Revistas ClubeDelphi, .NET Magazine Brasil e EASY .NET Magazine, professor de Pós-Graduação da UNIVEM-SP, na área .NET. Especialista em desenvolvimento Delphi, C#, .NET, ASP.NET, Visual Studio, atuando também em projetos de migração, orientando empresas e desenvolvedores na adoção de melhores práticas, ministra treinamentos em .NET focados em equipes Delphi. Foi arquiteto e desenvolvedor ASP.NET do site de doações on-line do projeto Criança Esperança da Rede Globo / Unesco.

Site oficial do Delphi Prism
http://www.embarcadero.com/products/delphi-prism

Wiki do Delphi Prism
http://prismwiki.codegear.com/en/Main_Page

Download do Delphi Prism
https://downloads.embarcadero.com/free/delphi_prism

Site oficial do ASP.NET
http://www.asp.net

Visual Studio
www.microsoft.com/visualstudio

Blogs do autor
http://twitter.com/guintherpauli
http://gpauli.com

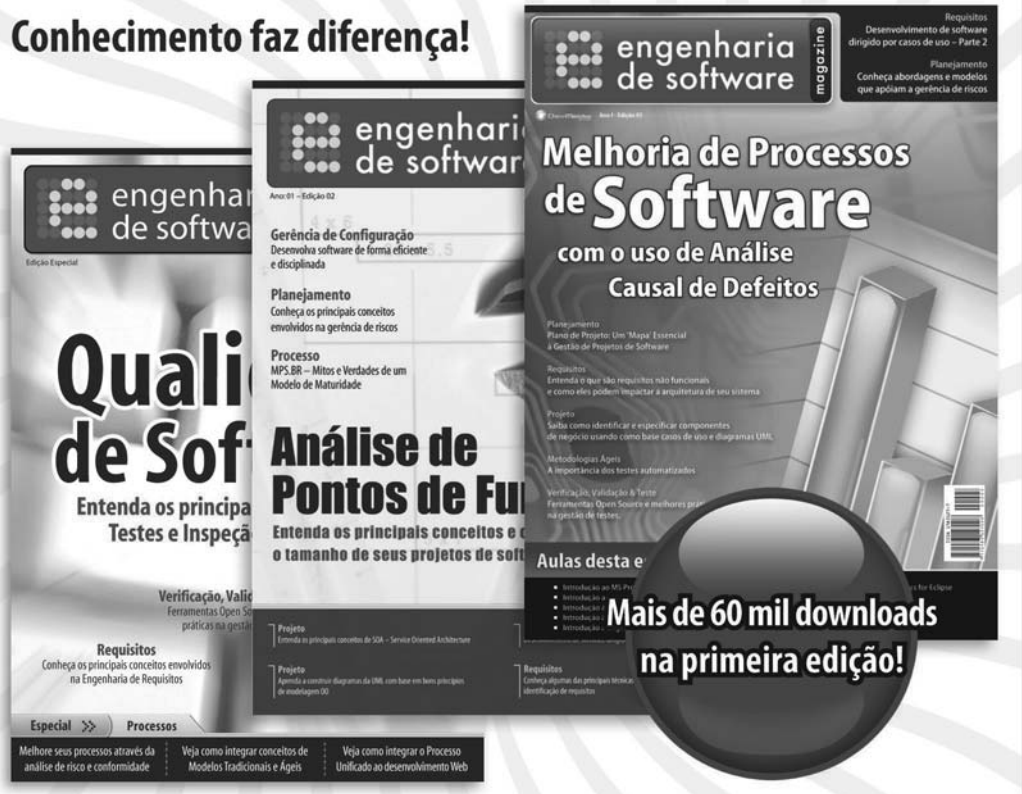
Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Conhecimento faz diferença!



Faça já sua assinatura digital! | *www.devmedia.com.br/es*

Faça um *up grade* em sua carreira.

Em um mercado cada vez mais focado em qualidade ter conhecimentos aprofundados sobre requisitos, metodologia, análises, testes, entre outros, pode ser a diferença entre conquistar ou não uma boa posição profissional. Sabendo disso a DevMedia lança para os desenvolvedores brasileiros sua primeira revista digital totalmente especializada em Engenharia de Software. Todos os meses você irá encontrar artigos sobre Metodologias Ageis; Metodologias tradicionais (*document driven*); ALM (*application lifecycle management*); SOA (aplicações orientadas a servicos); Analise de sistemas; modelagem; Métricas; orientação à objetos; UML; testes e muito mais. **Assine já!**

DevMedia group

Formatação de códig

Formate qualquer código-fonte a sua man

Resumo DevMan

De que trata o artigo:

O artigo trata de ilustrar maneiras de tornar seu código-fonte, que muitas vezes pode estar ilegível, em um código-fonte formatado, totalmente customizado ao seu gosto, permitindo o controle total e facilidade na leitura, entendimento e é claro manutenção do mesmo.

Para que serve:

Mostrar funcionalidades de aplicativos externos, ferramentas Open Source que podem auxiliar na adequação do seu código-fonte a maneira que mais lhe agrade, proporcionando padronização e facilidade de correção. Ao longo do artigo, conheceremos o Jedi Code Formater que nos proporciona este efeito e conheceremos também o utilitário de formatação de código embutido na nova versão do Delphi 2010.

Em que situação o tema é útil:

Em casos onde é necessário dar manutenção em códigos-fonte complexos e que já passaram por diversas mãos de diferentes desenvolvedores, considerando que cada um possui uma maneira de programar (endentar código, posicionamento dos blocos begin end, espaçamento entre outras particularidades), permite que se tenha uma padronização de código.

Formatação de código-fonte inteligente:

Conforme a dimensão de um software e seu ciclo de vida (considerando o tempo de desenvolvimento gasto até o produto atual) é provável que seu código-fonte passe pelas mãos de diversos desenvolvedores dos mais variados tipos. Embora tenhamos padrões de endentação de código, nem sempre estes são seguidos à risca, afinal, não existem "regras" para isso. Porém, conforme este crescimento esta falta de padronização pode vir a se tornar um problema, dificultando a manutenção, e consequentemente fazendo com que percamos tempo de desenvolvimento. Neste artigo iremos conhecer soluções que nos mostrem como contornar este problema de forma simples.

Quem trabalha a um tempo considerável com programação pode ser que tenha vivenciado experiência de dar manutenção em vários códigos-fontes. A endentação de código é a meu ver um fator básico para a qualidade da manutenção e leitura do código-fonte. Desde quando se inicia os estudos em programação, sempre ouvimos os professores nos orientarem sobre a importância da formatação, e a princípio não damos tanta importância para este fator, desconsiderando às vezes estes conselhos, porém com o decorrer do tempo, ganho de experiência, projetos complexos, códigos de outros desenvolvedores entre outros fatores, isto pode acabar sendo um problema. A princípio formatar o código é algo muito simples, porém o que fazer quando se possui um sistema muito complexo e com centenas de units? Você com certeza já deve ter passado por uma situação dessas, onde um código simples se tornava relativamente complexo devido à falta de organização e se não passou, prepare-se pois este dia pode chegar. Para este tipo de caso conheceremos a ferramenta Jedi Code Format.

O Jedi Code Format trata-se de um projeto Open Source, dos mesmos criadores da biblioteca JediVCL, que tem por objetivo a formatação de código-fonte tanto do tradicional Delphi quanto do Delphi na plataforma .NET. Com ele é possível realizar a formatação de espaçamento, por exemplo, definindo a quantidade de espaços entre a linha inicial, declaração de variáveis, constantes, parâmetros de procedures, blocos de início e fim (begin.. end), forma de endentação de código destes blocos, entre outras tantas configurações. Tudo isso pode ser feito através de customizações em um aplicativo externo (que possui código-fonte livre e disponível para acesso e estudo).

Nota

O link para download do Jedi Code Format encontra-se ao final deste artigo.

Faça o download do código-fonte e descompacte-o em uma pasta qualquer, você notará os executáveis JCFGui.exe e JCFNotepad. Irei comentar um pouco sobre ambos, porém não entrarei em grandes detalhes sobre a utilização da ferramenta, deixando o teste para o leitor que enquadrar-se nos casos anteriormente citados.

O JCFNotepad é um bloco de notas que permite inserir codificações do mesmo modo com que faríamos no Delphi, porém na aba Input, possibilitando posteriormente salvarmos este arquivo .pas. Ao realizar uma operação de abertura do arquivo, você notará que a Unit é carregada também na aba Input, permitindo já de imediato realizar a formatação através do botão "Go" e visualizar o resultado na aba Output, como mostra a **Figura 1**.

Basicamente o que temos no JCF Notepad seria um projeto de teste para conheceremos a funcionalidade da ferramenta. No menu Settings>Format Settings teremos a janela de customiza-

o-fonte inteligente

eira

Daniel Sobrinho Laporte

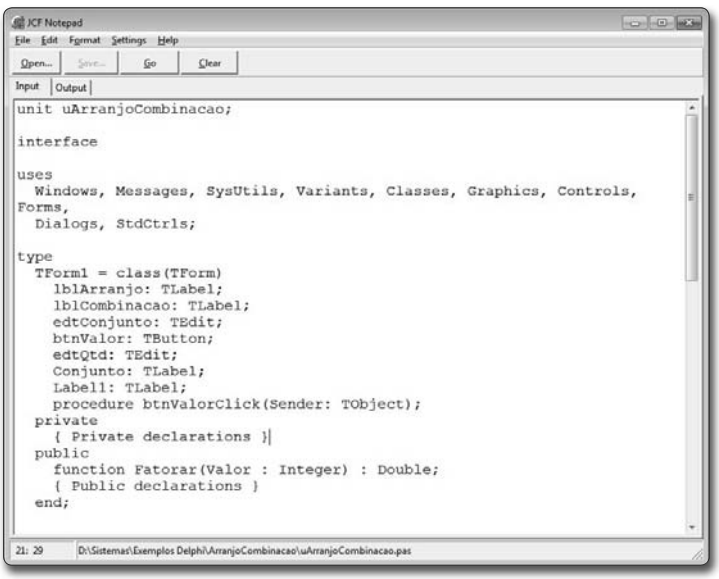


Figura 1. Carregando uma Unit no JCF Notepad

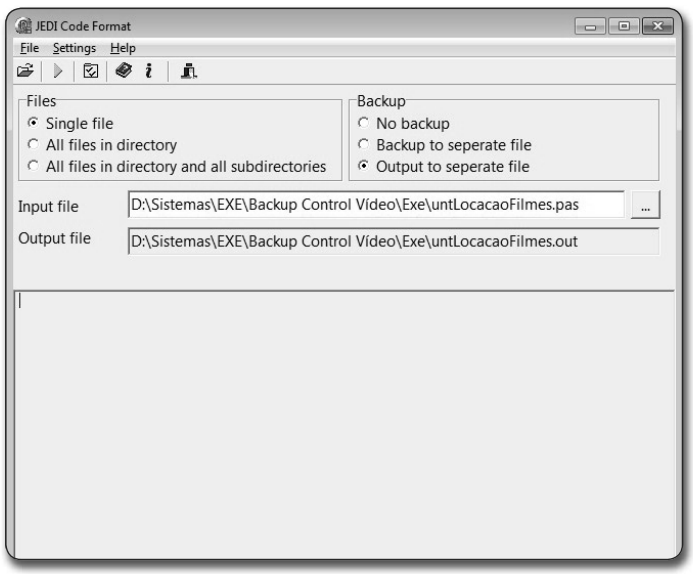


Figura 2. Janela do Jedi Code Format

ção das endentações, porém mostrarei esta tela mais adiante, pois a mesma tela está inclusa no projeto JCFGui.exe. Já que citamos o JCF Gui, execute-o. Você terá uma janela como a da **Figura 2**.

A janela da **Figura 2** ilustra a abertura do Jedi Code Format. Como você pode visualizar na imagem, o programa é relativamente simples. No grupo esquerdo temos a opção de realizar a formatação de um único arquivo, todos os arquivos de um diretório x ou todos os arquivos de um diretório x incluindo subdiretórios. No grupo direito temos a opção "No Backup" que irá sobrescrever o arquivo que estiver sendo informado, a opção de backup que salvará o arquivo de saída com uma extensão .bak e por último a opção de saída de output que irá gerar o arquivo com a extensão .out. Para realizar a formatação basta clicar no botão Run representado pelo botão de play na parte superior. Assim como no projeto anterior, temos aqui a opção de formatação no menu Settings>Format Settings conforme mostra a **Figura 3**.

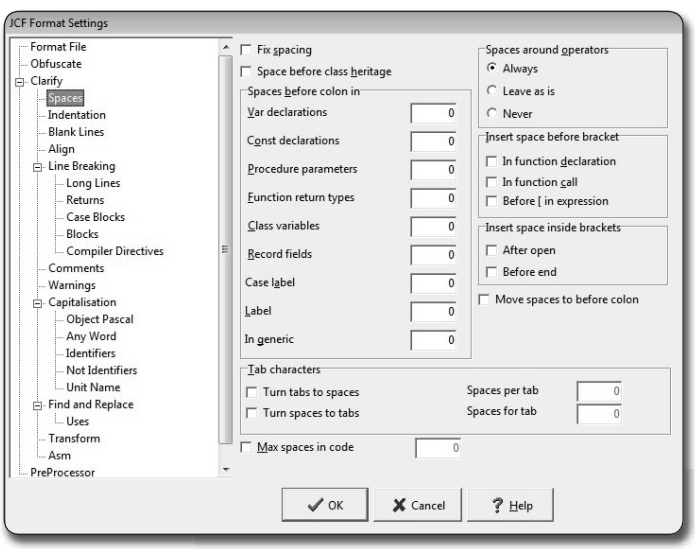
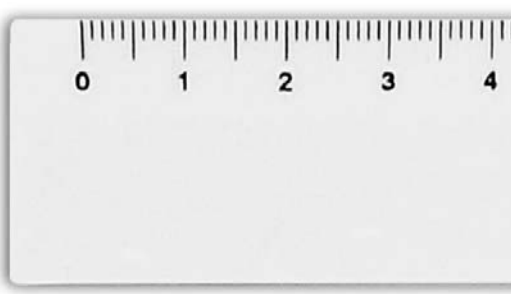


Figura 3. Janela de customização de formatação

Note no TreeView do lado esquerdo que temos a opção Spaces selecionada e dentro dela a customização de quantidade de espaços por variáveis, constantes, parâmetros de procedure, resultados de funções, variáveis de classe e configuração até dos espaçamentos dados pela tecla Tab, em Tab characters. Note também no TreeView que ainda temos a customização de endentação (indentation), alinhamento, blocos case, diretivas de compilação, remoção de comentários e muito mais opções. Explorando o site do projeto você verá que no próprio rodapé da página são informados outros projetos



para esta mesma finalidade. Há também uma .bpl que permite a integração com o Delphi, embora não na versão 2010, porém pode ser facilmente adaptada para o mesmo.

Se você já utiliza o Delphi 2010, e está com dificuldades de migrar esta .bpl ou simplesmente não deseja ter este trabalho, não pense que não é possível realizar isso na versão 2010. É possível realizar a configuração da endentação de código e inclusive utilizar os recursos de formatação automática do código assim como o Jedi Code Format. Veremos como utilizar estes recursos diretamente no seu IDE.

Abra o Delphi 2010 e crie uma nova aplicação em File>New>VCL Forms Application – Delphi e logo de início, pressione a tecla F6 para chamar o IDE Insight. Na janela do IDE Insight informe a palavra "Formatter" como mostra a **Figura 4**.

Observando a **Figura 4** podemos notar uma duplicidade nas opções Indentation page, Line breaks e Spaces page. Essa duplicidade se deve é claro, as opções para a utilização do C++, pois no mesmo IDE é possível configurar os dois ambientes. Caso você queira acessar esta aba sem que seja pelo IDE Insight utilize o menu Tools>Options>Formatter e expandindo esta opção teremos as opções que aparecem no IDE Insight, conforme mostra a **Figura 5**.

Vamos explorar um pouco as opções de configuração, conhecer essas opções e por último fazer o teste prático. Portanto, selecione dentro da opção Delphi o subitem Indentation. Dentro desta opção temos tudo relacionado à endentação do código-fonte. É possível configurar nesta opção a quantidade de linhas que devem ser consideradas em uma endentação como em um código como a seguir:

```
If 1 = 1 then ShowMessage('Verdadeiro') else
ShowMessage('Falso');
```

Particularmente este é um exemplo do que você não deve fazer, mas como nem sempre o código-fonte vai ser feito por você, então pode acontecer de ver algo como o exemplo anterior. Neste caso o item Continuation indent (marcado como 2) endentaria o código com dois espaços após o then, ficando:

```
If 1 = 1 then
  ShowMessage('Verdadeiro')
else
  ShowMessage('False);
```

Além desta opção temos como determinar as opções para desconsiderar a endentação após a posição x (configurada como 40 por padrão), endentação dos blocos begin / end, endentação dos blocos de código entre o begin e o end, definições dos corpos de classes, comentários, diretivas de compilação, endentação de seções (implementation, interface). Há também a possibilidade de personalização dos blocos case e endentação por labels, considerando a posição esquerda para o alinhamento.

Na opção Spaces é possível determinar opções relativas aos espaçamentos do código, como comentários, operadores, parênteses entre outros.

Line Breaks permite customizar tudo relacionado a opções de quebra de linhas, possuindo opções formas de formatar ao inserir quebras de linha, ao inserir quebras de linha após o begin e uma personalização de números de linhas em branco. Por último temos aqui uma novidade chamada de "Capitalization", na qual permite transformar os primeiros caracteres minúsculos em maiúsculos. Veja a tela na **Figura 6**.

Não irei modificar nenhuma opção, deixando tudo como padrão. A seguir mostraremos como aplicar o recurso de formatação. Para isso no novo formulário que criamos anteriormente dê um duplo clique, desta forma iremos codificar o evento OnCreate do formulário conforme vimos anteriormente. Veja na **Figura 7**.

Para aplicar a formatação podemos selecionar o código e pressionar simultaneamente as teclas Ctrl + D ou pelo botão direito do mouse na opção Format Source, como mostra a **Figura 8.**

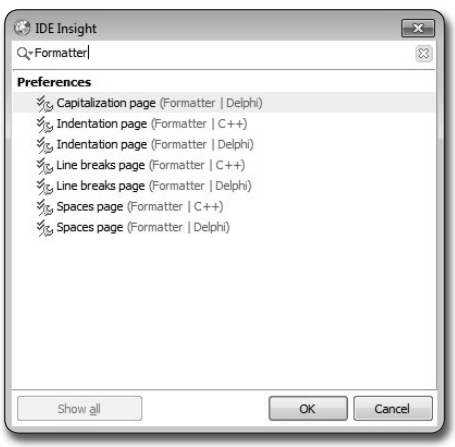


Figura 4. Encontrando o formatador via IDE Insight

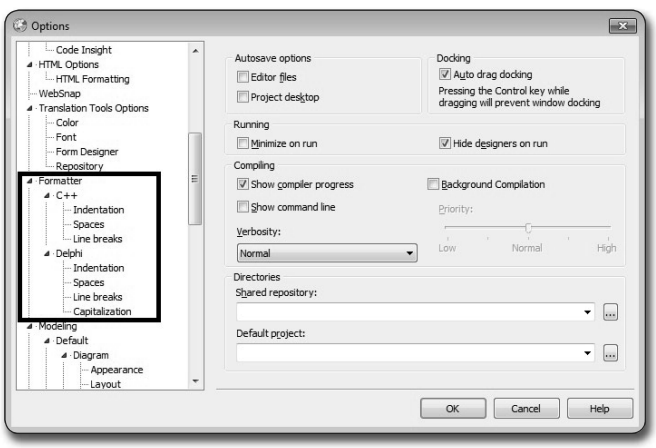


Figura 5. Exibindo as opções de configuração de formatação

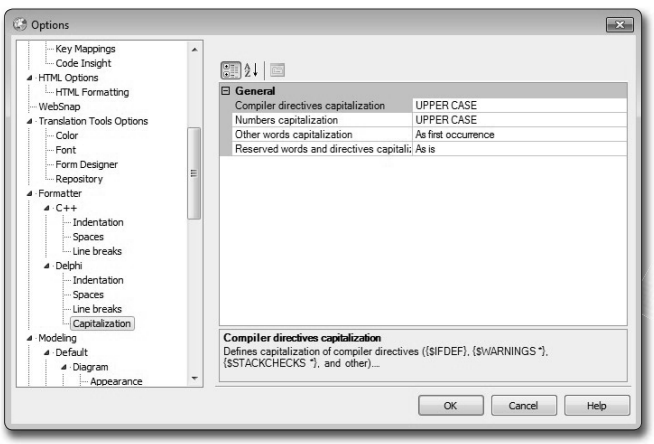


Figura 6. Novo recurso denominado "Capitalization"

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  if 1 = 1 then ShowMessage('Verdadeiro') else ShowMessage('Falso');
end;
```

Figura 7. Evento OnCreate simulando uma formatação errada

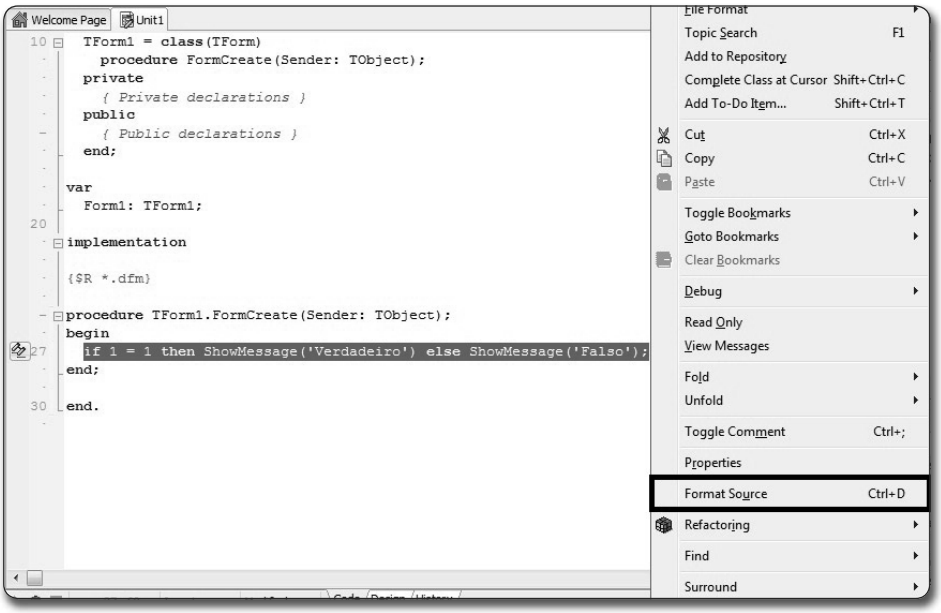


Figura 8. Exemplo de aplicação do recurso de formatação de código do Delphi 2010

Após a aplicação do recurso o seu código-fonte ficará completamente endentado, como na **Figura 9**.

Conclusão

Além destas opções que vimos no artigo existem outras ferramentas que proporcionam o mesmo efeito, como o excelente add-in Castalia, que possui muitos outros recursos que vão além da endentação de código. Com tudo isso vimos que agora não há desculpas para ter um código desorganizado. Vale lembrar que essa prática de código formatado é uma boa prática, pois pode tornar sua manutenabilidade de código muito mais fácil e é claro, simples. Um grande abraço e até a próxima!

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  if 1 = 1 then
    ShowMessage('Verdadeiro')
  else
    ShowMessage('Falso');
end;
```

Figura 9. Código após identação

Daniel Sobrinho Laporte

daniel.sobrinho@devmedia.com.br
daniel.laporte@gmail.com
www.twitter.com/daniel_laporte

Colunista e editor técnico da Revista ClubeDelphi. Graduado em Sistemas de informação e pós-graduado em Administração de Bancos de Dados pela Universidade de Franca. É analista de sistemas e desenvolvedor Delphi desde 2005 desenvolvendo aplicações cliente/servidor e multicamadas.

Jedi Code Format
http://jedicodeformat.sourceforge.net/

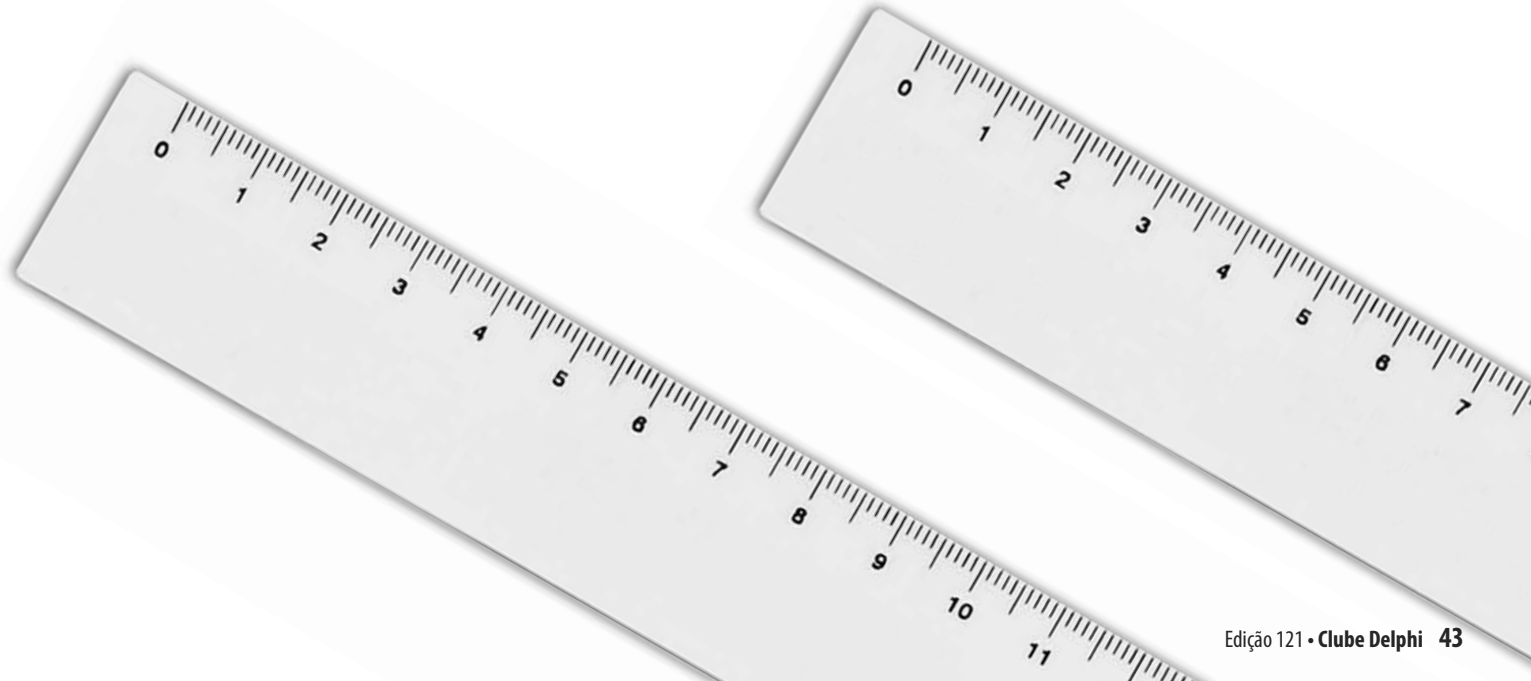
Castalia
http://www.twodesk.com/castalia/

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback



Twitter, jQuery e P

Integrando seu Web Site com o Twitter e

Dificilmente encontramos algum internauta que não faça parte de alguma rede social. Um fenômeno que tomou conta a nível global e vem se difundindo cada vez com maior intensidade. As redes sociais tornaram-se um mecanismo de comunicação muito valioso para qualquer pessoa e ainda de forma bastante intensa com o meio empresarial.

Diariamente pequenas e grandes empresas, sempre tão distantes de seu consumidor, importam-se cada vez mais com este meio de comunicação, visto que, uma simples opinião pode pôr abaixo todo um planejamento de marketing ou ainda um produto, mas em contra partida, pode ser um canal de comunicação valioso, ajudando a difusão de informações, campanhas e interação com seus clientes.

Uma das redes sociais que mais tem tido destaque atualmente é o Twitter. Esta rede social com seus 140 caracteres tomou conta de uma fatia muito grande de internautas interessados em informações rápidas e instantâneas. Mas, esta rede social vai muito além, possibilitan-do aos usuários uma comunicação sem igual com seus ídolos, empresas e ainda seus amigos. Mensagens das mais variadas correm em sua tela podendo ser elas até uma nova vaga de emprego ou um produto com um valor de ocasião, ou você não se interessa quando o alto-falante do supermercado anuncia uma promoção relâmpago?

Esta é uma nova realidade e devemos estar preparados. Podemos proporcionar aos usuários de nossos sistemas algumas facilidades quanto a estes serviços, sendo que, praticamente todos eles, disponibilizam aos desenvolvedores as suas APIs (Application Programming Interface) onde é possível que seu sistema e a rede social em questão se comuniquem.

Resumo DevMan

De que trata o artigo:

Este artigo mostrará ao leitor como integrar tecnologias para utilização da rede social Twitter. Através do exemplo criado o leitor criará um sistema de envio de mensagem com foto ao Twitter e recuperando o nome da foto através do serviço do twitpic.com usando para isso as tecnologias Ajax com jQuery e PHP.

Para que serve:

Através do projeto criado, o desenvolvedor terá condições de implantar em seus sistemas Web recursos de integração com o Twitter, como por exemplo, ao cadastrar um produto enviar uma mensagem e uma foto do mesmo para todos os seus seguidores.

Em que situação o tema é útil:

Este assunto é útil em situações onde o desenvolvedor deseja facilitar a vida de seu usuário trazendo para dentro de seu próprio sistema a comunicação com a API do Twitter, por exemplo, uma loja que ao cadastrar um novo produto em seu site já tenha a possibilidade de enviar uma foto ao Twitpic e uma mensagem para todos os seus seguidores conhecerem este novo produto.

Twitter, jQuery e PHP:

Vamos aprender a utilizar a biblioteca jQuery e seus recursos de Ajax. Através de um exemplo prático, vamos aprender a criar a página HTML básica e através de JavaScript e PHP criaremos todos os comportamentos necessários para que nossa página realize o envio de uma imagem, via Post, para a API do Twitpic, enviando a imagem e uma mensagem para todos os seguidores do usuário do sistema.

Estes serviços são, em linhas gerais, gratuitos e de fácil utilização. Através de poucas linhas conseguimos uma comunicação satisfatória de nossa página com o serviço do Twitpic por exemplo. Este serviço possibilita ao internauta o envio de fotos e ainda envio de mensagens aos seus seguidores, de forma gratuita e relativamente simples, bastando que você possua uma conta no Twitter.

A biblioteca jQuery

Escrever código em JavaScript não é uma das tarefas mais fáceis que podemos executar. A incompatibilidade entre Browsers ou ainda a forma como cada um interpreta o código também é conhecida, ou seja, um código que escrevemos e funciona no Firefox pode vir a não funcionar no Internet Explorer, e vice versa. Neste contexto nascem as bibliotecas. Estes elementos são arquivos contendo códigos já prontos que visam facilitar e trazer recursos mais eficientes e ainda fazer com que tenhamos de digitar menos e obter mais resultados. jQuery nada mais é do que uma dessas bibliotecas.

A biblioteca jQuery (ou Projeto jQuery) foi inicialmente desenvolvida por John Resig e teve sua primeira versão disponibilizada em agosto de 2005, onde contemplava uma pequena gama de recursos. A ideologia por trás deste projeto é facilitar o desenvolvimento de scripts utilizando JavaScript e ainda facilitar a resolução de problemas relacionados a diferenças entre Browsers (Cross Browser). A simplicidade e sua curva de aprendizado são bastante relevantes na utilização desta biblioteca, pois, para quem já possui uma base de

HP Twitpic

Jaison Schmidt

JavaScript, não precisará de muito tempo até se adaptar e entender seus conceitos.

O termo Ajax, para muitos, pode ser encarado como um enorme problema. Este é sem dúvida uma das grandes facilidades trazidas pela biblioteca jQuery, onde com poucas linhas de código e métodos prontos, podemos invocar recursos avançados e obter resultados bastante confiáveis.

Outra facilidade por trás da biblioteca jQuery é a utilização de Plugins, ou seja, podemos baixar complementos à biblioteca que estendem suas funcionalidades, facilitando a utilização de recursos avançados ou que seriam de dificuldade mais elevada, mesmo utilizando a biblioteca em sua forma nativa. Alguns Plugins acabam por se destacar através de suas funcionalidades, como por exemplo, estão disponíveis Plugins para gerenciamento de formulários, onde com poucos métodos podemos enviar um formulário inteiro via Ajax e tratar seu retorno da forma como for necessária. Existem algumas extensões que trazem recursos de Layout, como por exemplo, recursos de Drag and Drop (arrastar e soltar) e/ou ainda recursos de janelas modais e muitos outros recursos interessantes e de utilização bastante simplificada.

Entendo APIs

Para desenvolvedores iniciantes, este termo pode soar como algo bastante distante e confuso. Uma API (Application Programming Interface) é a forma que podemos utilizar para fazer a comunicação com outros sistemas, um exemplo simples, você deseja postar uma mensagem no Twitter através de seu próprio sistema, como proceder? Utilizando a API do Twitter, é claro.

Em nosso artigo faremos uso de um serviço chamado Twitpic, que é um repositório de imagens integrado ao Twitter. Através deste serviço podemos enviar uma foto para uma URL específica e receber como retorno várias informações, e toda esta comunicação é feita através de uma API.

Na grande maioria dos casos, em se tratando de Web, a comunicação dos dados se dá via XML, mas isto não é obrigatório. Existem várias formas de sistemas se comunicarem e trocarem informações, uma delas é utilizando XML ou ainda JSON, o Twitpic nos permite trabalhar com estes dois padrões de dados, ficando a nosso critério qual nos agrada mais, em nosso exemplo utilizaremos XML.

O serviço Twitpic requer uma conta no Twitter. Se você já possui um usuário neste serviço você pode utilizar o seu usuário e senha, se não possui cadastro, acesse o Twitter e crie sua conta. No decorrer de nosso artigo veremos as principais formas de comunicação com o Twitpic, quais URLs podemos utilizar e ainda o tipo de retorno gerado pelo serviço.

Criando nosso exemplo

Antes de iniciarmos o desenvolvimento, devemos ter os objetivos bem claros. O que desejamos afinal? Criar uma tela onde o nosso cliente selecione uma foto em seu computador informe uma mensagem e, feito isso, utilizando Ajax e sem recarregar a página, enviar a foto e a mensagem através do serviço do Twitpic. Como um Plus, podemos tratar o retorno disso, capturando o nome da imagem e o endereço para em um futuro acessarmos a mesma, gravarmos este nome em um banco de dados ou reutilizar esta imagem.

O primeiro passo consiste em organizarmos o ambiente de trabalho. Vamos precisar de um servidor Web, este pode ser local ou remoto, em nosso exemplo utilizaremos um servidor local, mais especificamente o pacote XAMPP que é gratuito e possui todos os recursos necessários ao desenvolvimento de nosso exemplo. Este pacote possui um servidor Web (Apache) e os módulos PHP já prontos e configurados.

Como editor de código podemos utilizar qualquer editor de textos, visto que não necessitaremos de nenhum recurso especial fornecido por ambientes de desenvolvimento. Atualmente temos muitos editores de código, desde o simples bloco de notas até os mais robustos como, por exemplo, Dreamweaver, Visual Studio e Netbeans, que suporta códigos em PHP e é uma ótima alternativa gratuita para programadores de PHP, embora o mesmo seja velho conhecido da comunidade Java.

Nota do DevMan

Existem diversos pacotes de servidores Web gratuitos. O XAMPP é bastante conhecido entre desenvolvedores PHP. Você pode também instalar o servidor Apache e configurar todos os módulos e recursos manualmente, mas os pacotes facilitam muito a vida do desenvolvedor. A instalação (para usuários do sistema operacional Windows) é bastante simples e trivial, bastando baixar o pacote (link disponível na sessão Links ao final deste artigo) e executar o instalador. Por padrão o XAMPP é instalado em "C:\xampp" e os arquivos de nossos sites ficam em "C:\xampp\htdocs", podendo ser acessados via URL em "HTTP://localhost/nomedapasta". Após o procedimento de instalação, na sua área de trabalho deve haver um ícone com o nome XAMPP Control Panel. Dê dois cliques em cima do mesmo e clique em "Start" ao lado da palavra Apache, isso inicializará o nosso servidor Web. Para visualizar o servidor em execução você deve acessar http://localhost/ e a tela inicial do XAMPP deve abrir, caso isto não aconteça, verifique as configurações utilizadas.

Entendendo o problema

Nosso cliente deseja enviar uma foto de seu produto e uma mensagem via Twitter para todos os seus seguidores. Como já dito, isto poderia ser feito em sistemas de e-commerce, mas, podemos programar em qualquer site que necessite de integração com redes sociais. Em sites pessoais, por exemplo, pode ser um bom diferencial para conquistar o cliente e vender o seu projeto.

Nosso exemplo será uma aplicação com um formulário HTML, um serviço em PHP e a interligação feita através de JavaScript, sendo que este último utilizará os recursos da biblioteca jQuery e Ajax. Na **Figura 1** podemos ver um esquema de funcionamento desta pequena aplicação. Lembre-se: O principal não é saber fazer e sim entender de forma correta o que está sendo feito para que, quando necessário, você saiba onde procurar a informação e ainda os conceitos necessários na resolução de determinado problema.

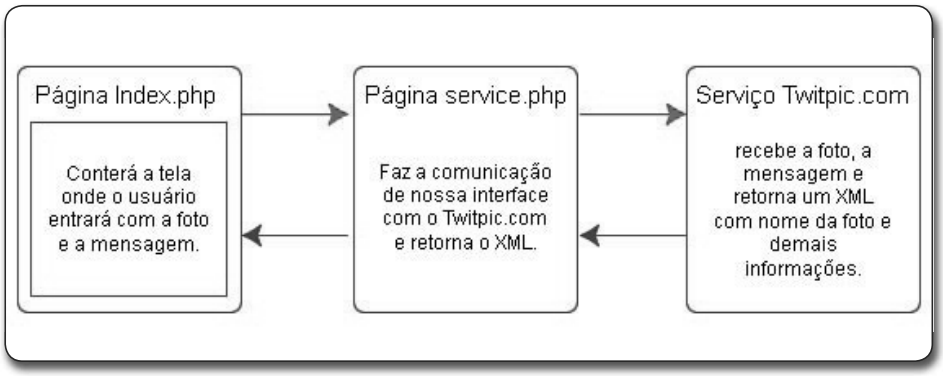


Figura 1. Esquema de funcionamento do projeto

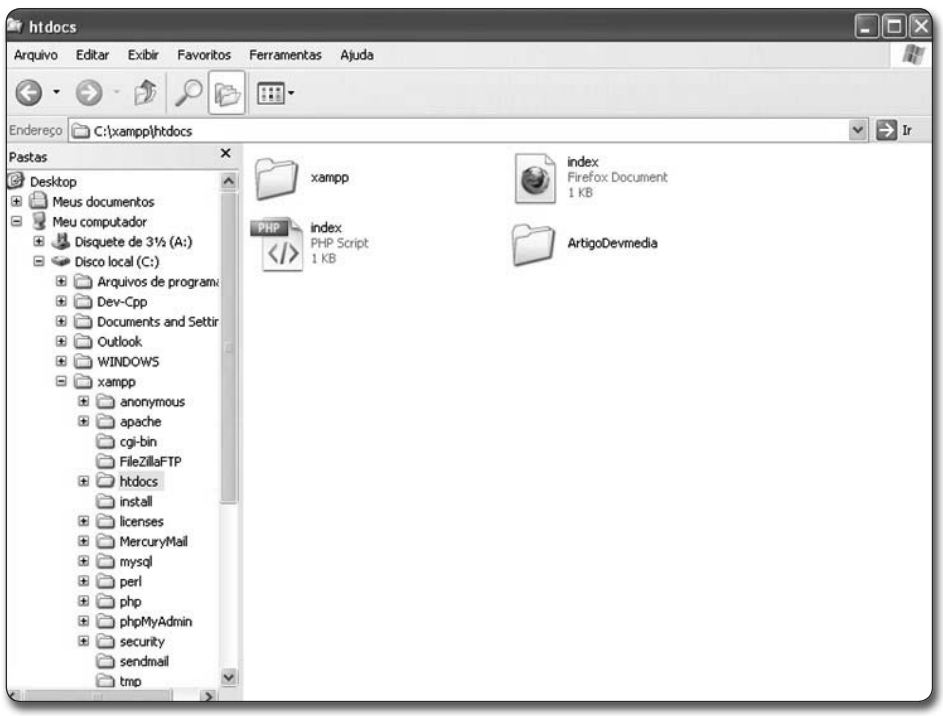


Figura 2. Estrutura das pastas utilizadas em nosso projeto

Codificação da página index.php

Considerando que nosso servidor já esteja funcionando corretamente, vamos criar a pasta "ArtigoDevmedia" na pasta c:\xampp\htdocs conforme mostra a **Figura 2**.

Nossa primeira página será chamada de index.php, como já falado anteriormente, você pode utilizar qualquer editor de textos para a codificação, em nosso exemplo utilizaremos o próprio bloco de notas.

Esta página é a estrutura HTML, ou seja, a interface com nosso usuário. É através dessa página que o nosso cliente irá localizar a imagem em seu computador e ainda preencher a mensagem que será postada junto a nossa conta do Twitter.

Esta página possuirá uma sintaxe bastante simples, visto que será composta de apenas um campo de imagem e outro campo de texto para a mensagem que nosso cliente irá digitar, como podemos ver na **Listagem 1**.

Para quem já tem familiaridade, o código da **Listagem 1** é bastante simples, sendo um HTML simples com os elementos básicos de um formulário, então vamos nos ater às particularidades. Perceba que

Listagem 1. Index.php página inicial

```
01 <html xmlns="http://www.w3.org/1999/xhtml">
02 <head>
03 <title>jQuery com Twitter</title>
04 <script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.3/jquery.min.js"></script>
05 <script type="text/javascript" src="jquery.form.js"></script>
06 <script type="text/javascript" src="mainJS.js"></script>
07 </head>
08 <body>
09 <div id="retorno"></div>
10 <form action="service.php" method="post" name="frmSend" id="frmSend">
11 <p><label id="lblImagem">Selecione a imagem:</label>
12 <input type="file" id="cpFoto" name="cpFoto" /></p>
13 <p><label id="lblMsg">Escreva sua mensagem:</label>
14 <input name="txtMsg" type="text" id="txtMsg" maxlength="140" /></p>
15 <input type="submit" value="Enviar" id="btnEnv" />
16 </form>
17 </body>
18 </html>
```

todos os elementos dentro da tag <body> possuem um ID, ou seja, são únicos e, portanto facilmente identificáveis. Outro item relevante encontra-se na linha 04 onde temos a inclusão da biblioteca jQuery, que encontra-se no repositório do Google Code, dessa forma você não precisa baixar para o seu servidor, economizando um pequeno espaço e algum tráfego.

Na linha 05 temos a inclusão do arquivo "jquery.forms.js". Este arquivo é um Plugin para a biblioteca jQuery e em consequência estende as funcionalidades da mesma. Esta biblioteca nos fornece métodos bastante simples e eficazes de trabalharmos com formulários utilizando Ajax, onde com poucas linhas podemos fazer o envio e recebimento de informações sem a necessidade de recarregar a página. Este Plugin está disponível junto com o código fonte final deste exemplo ou ainda através do endereço citado na sessão de Links.

Na linha 06 temos a inclusão do arquivo "mainJS.js" que ainda não foi criado, e será neste arquivo que vamos programar em JavaScript. Perceba que na linha 10 o nosso formulário possui a propriedade action="service.php" sendo que esta página ainda não foi criada, mas será a página que fará a comunicação de nosso formulário com o serviço do Twitpic, enviando as informações de usuário, senha, mensagem e foto e retornando um XML com as informações de postagens.

Nas linhas 12 e 14 temos a definição dos campos de imagem e mensagem, estes são campos padrão HTML e não devem apresentar problemas. Tome bastante cuidado com os atributos ID, lembrando que o JavaScript é Case Sensitive, ou seja, você deve cuidar as letras maiúsculas e minúsculas.

Após digitar este código no bloco de notas ou em seu editor preferido, salve-o com o nome de "index.php" na pasta C:\xampp\htdocs\ArtigoDevmedia (considerando que você tenha instalado o servidor XAMPP na pasta c:\xampp). No bloco de notas você deve colocar o nome entre aspas e selecionar "Todos os arquivos" conforme podemos ver na **Figura 3**.

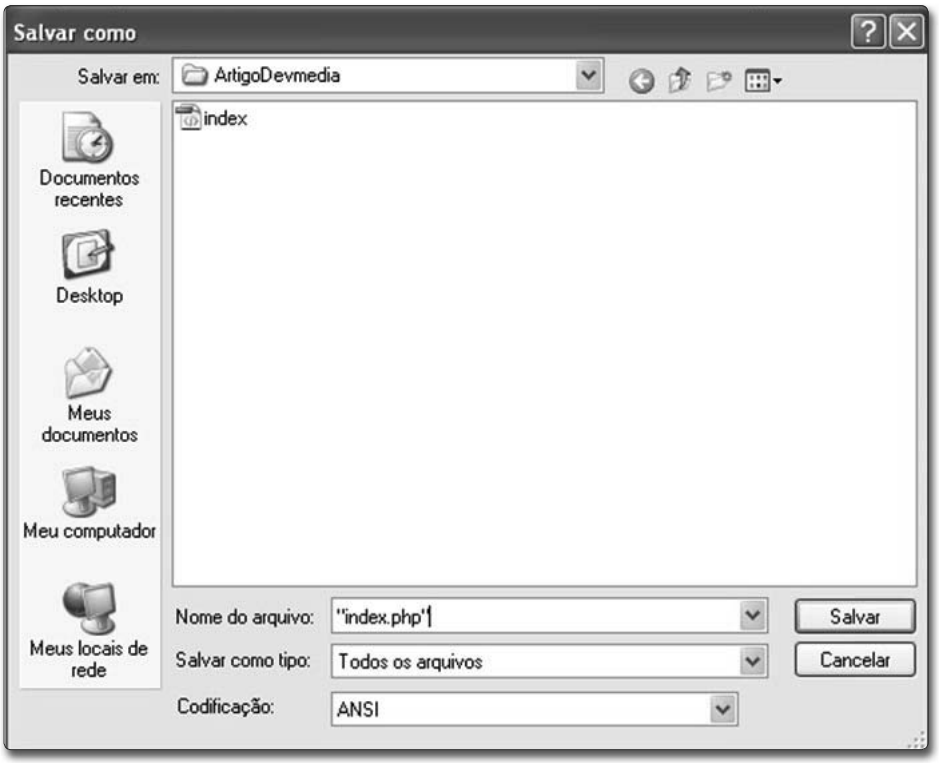


Figura 3. Padrões de como deve ser salvo o arquivo

Após salvar, você pode acessar o endereço http://localhost/ArtigoDevmedia/ e nossa página recém criada aparecerá, somente com o formulário, e nenhuma operação é realizada após o preenchimento dos campos e o envio do formulário.

Codificando o arquivo mainJS.js

Vamos agora criar o nosso arquivo de script mainJS.js. Este nome não é obrigatório, você pode utilizar outro de sua escolha, mas mude o nome na linha 06 de nosso arquivo index.php. Neste novo arquivo faremos toda a criação de nosso Script que realizará as operações utilizando jQuery. O código deste arquivo pode ser visualizado na **Listagem 2**.

A ideia por trás deste arquivo é bastante simples. Vamos verificar se os campos estão preenchidos, se não estiverem, informamos e paramos a sua execução. Caso estejam preenchidos, vamos enviar o nosso formulário para a página "service.php", utilizando Ajax. Durante este processo, informaremos ao usuário que estamos enviando o formulário e, quando obtivermos o devido retorno, mostraremos a imagem na tela e retornar o endereço no serviço Twitpic para visualizar a mesma.

Crie um novo arquivo no bloco de notas, insira o código da **Listagem 2** e salve o mesmo com o nome de "mainJS.js" na mesma pasta em que está o nosso arquivo index.php.

O código apresentado na **Listagem 2** pode parecer bastante complicado para desenvolvedores iniciantes, mas não chega a ser tão complexo. A biblioteca jQuery traz grande facilidade na codificação de atividades mais complexas do JavaScript. O que está escrito entre as linhas 02 e 18 será executado quando a página for carregada, este bloco é equivalente à propriedade window.load do JavaScript normal, em jQuery esta função é descrita como na linha 01, ou seja, jQuery(function(){ // código });. Tudo o que está entre as chaves ({}) será executado quando o documento carregar. Os métodos da biblioteca jQuery começam justamente por jQuery(seletor), ou o seu equivalente $(seletor). Perceba

Nota

As bibliotecas utilizadas neste artigo são gratuitas e de código aberto. Pode-se utilizar estes recursos com as mais diferentes linguagens de programação, inclusive .NET e Java, não ficando sua utilização restrita somente a aplicações em PHP.

que podemos utilizar ambas as sintaxes, mas optamos por jQuery(seletor) para não ocorrerem conflitos com outras bibliotecas que também utilizem o coringa $(seletor), visto que esta última não é exclusividade da jQuery.

Quando falamos em seletor estamos falando do objeto que está sendo selecionado. O seletor é um equivalente a funções do tipo "getElement", pois, através da expressão contida, podemos selecionar um ou mais elementos em nossa página. Não fique preocupado se não entendeu tudo que foi escrito, com os exemplos vamos entender bem melhor.

Nas linhas 04 e 05 capturamos os objetos (campos) cpFoto e txtMsg, se você perceber utilizamos o seletor jQuery("#ID_ELEMENTO") para capturar o campo, esta expressão é equivalente a document.getElementById("ID_ELEMENTO") e nos retorna o objeto apontado pelo ID. Na linha 07 definimos a função click para o nosso botão de envio do formulário, ou seja, quando clicarmos no botão "btnEnv" vamos disparar esta função.

Quando clicamos no botão btnEnv, a função click é disparada, e dentro desta função fazemos a verificação de preenchimento dos campos, ou seja, verificamos se os campos estão preenchidos. Em uma situação real, deveríamos verificar os espaços em branco antes e depois do preenchimento do campo e a validade dos conteúdos, mas como o nosso intuito é verificar a comunicação com o serviço Twitpic, não faremos grandes validações.

Na linha 14 definimos a variável options, que utiliza estrutura JSON para armazenar os parâmetros a serem passados ao método ajaxForm, presente na linha 15. Este método é implementado pelo Plugin jquery.form.js que estende as funcionalidades do objeto jQuery. A variável options armazena algumas informações, como por exemplo, o tipo de resposta que vamos obter (dataType: 'xml'), a função que será chamada durante o processamento da chamada (beforeSubmit: showRequest) e a função que será chamada quando obtivermos o retorno da chamada (success: processaXml).

Listagem 2. mainJS.js Arquivo de script contendo as funções de ajax e jQuery

```
01  // Função idêntica a window.load
02  jQuery(function(){
03  // captura os campos de foto e mensagem
04   var campoFoto = jQuery("#cpFoto");
05   var campoMsg = jQuery("#txtMsg");
06  // cria a função de submit para o formulário
07  $("#btnEnv").click(function(event) {
08  // apenas verifica se os campos estão preenchidos
09  if((campoFoto.attr('value') == "") && (campoMsg.attr('value') == "")) {
10       alert("Selecione uma imagem e preencha o campo da mensagem");
11       return 0;
12   }
13  // seta os parâmetros para o objeto ajaxForm
14  var options = { dataType: 'xml', success: processaXml, beforeSubmit:  showRequest };
15  // faz o envio do formulário via ajax, portanto, sem recarregar a página
16  jQuery('#frmSend').ajaxForm(options);
17    });
18   });
19   // Função que é executada enquanto o formulário está sendo processado
20   function showRequest() {
21    jQuery("#retorno").html("Enviando...");
22   }
23   // Função executada após o retorno gerado pela página service.php
24   function processaXml(responseXML) {
25  // limpa o texto "Enviando..."
26  jQuery("#retorno").html("");
27  // inicia o tratamento do retorno, que se dá via XML
28  jQuery(responseXML).find("rsp").each(function(){
29  // captura o atributo que indica se o envio funcionou ou não
30      var lnk =  jQuery(this).attr("stat");
31  // se for fail, avisa que falhou e mostra a mensagem
32      if(lnk == "fail") {
33          jQuery(responseXML).find("err").each(function(){
34          var msg = jQuery(this).attr("msg");
35      alert('Falha no envio. \n'+msg);
36       });
37       } else {
38       // o envio da imagem funcionou, exibimos a imagem na tela
39       // a variável img armazena o nome da imagem gerada pelo Twitpic
40       var img = jQuery(responseXML).find("mediaid").text();
41      jQuery('#retorno').html('Imagem enviada com sucesso!<p><img src="http://twitpic.com/show/
        mini/'+img+'" /></p><p>Endereço da imagem: http://www.twitpic.com/'+img+'</p>');
42   }
43  });
44 }
```

Nota

O Plugin jquery.form.js deve ser baixado e colocado na mesma pasta do projeto. O Link para download você encontra na sessão Links, ao final deste artigo.

A função showRequest(), presente na linha 20 é bastante simples e apenas escreve dentro da div com id "retorno" o texto "Enviando..." para que o usuário saiba que algo está sendo feito.

A função processaXml (responseXML) é um pouco mais complexa e requer um pouco mais de cuidado. Ela recebe por parâmetro o XML que retorna do servidor Twitpic, ou seja, enviamos a imagem, a imagem foi cadastrada e o Twitpic nos envia um XML contendo uma série de informações, a estrutura do XML de retorno pode ser vista na **Listagem 3**.

Listagem 3. Exemplo de XML de retorno do serviço Twitpic.com

```
<?xml version="1.0" encoding="UTF-8" ?>
<rsp stat="ok">
    <mediaid>1xtkhi</mediaid>
    <mediaurl>http://twitpic.com/1xtkhi</mediaurl>
</rsp>
```

O jQuery interpreta arquivos XML da seguinte forma: você busca um determinado campo, como na linha 28, quando encontra, você pode recuperar seus atributos, como é feito na linha 30, onde localizamos

o atributo stat, que se for ok, indica que a foto foi postada com sucesso, se for fail significa que houve um erro. Quando houver erro, teremos a presença do atributo msg, que é recuperado na linha 34. Se tudo correu de forma satisfatória, localizamos o campo mediaid que está dentro do campo rsp, este campo armazena o código de nossa imagem no serviço Twitpic, perceba a presença do código jQuery(responseXML).find("mediaid").text(); que captura o texto dentro do campo, e não o valor de um atributo.

Por fim, na linha 41 montamos o retorno para o nosso usuário, onde o mesmo pode ver a miniatura de sua imagem e ainda obter um Link para esta imagem. Lembre-se que JavaScript é Case Sensitive, ou seja, diferencia maiúsculas e minúsculas, e ainda tome cuidado com os ";" e abertura e fechamento de aspas, grande parte dos erros em scripts estão em pequenos detalhes que, muitas vezes, são de difícil identificação.

Codificação do arquivo service.php

O arquivo service.php é quem faz o "meio de campo" entre a nossa interface em HTML e JavaScript e o serviço do Twitpic. Neste arquivo é que centralizamos o envio e recebimento das informações entre os serviços, ou seja, recebemos a foto e a mensagem que desejamos enviar, após, efetuamos o envio utilizando a biblioteca CURL do PHP.

Essa biblioteca tem por finalidade obter informações de outros sites. Existem mais formas de realizar esta operação, mas a biblioteca CURL nos dá uma série de funcionalidades bastante interessantes conforme podemos ver na **Listagem 4.** Crie um novo arquivo no bloco de notas, insira o código da **Listagem 4** e salve-o com o nome de "service.php" na mesma pasta em que está o nosso arquivo index.php.

Na **Listagem 4** podemos ver o recurso CURL em funcionamento. Da mesma forma como uma série de bibliotecas, podemos definir uma série de propriedades para o recurso antes de executar o mesmo, ou seja, antes definimos tudo que ele deve fazer ou como deve se comportar e, após estas definições, executamos o recurso.

Listagem 4. Service.php - Arquivo contendo as funções de comunicação entre os serviços

```
01 <?php
02    // cria o array que armazenará as informações
03    $info = array();
04    // armazena o nome de usuário do twitter
05    $info['username'] = "seu_usuario_twitter";
06    // armazena a senha deste usuário
07    $info['password'] = "sua_senha_twitter";
08    // armazena a mensagem
09    $info['message'] = $_POST['txtMsg'];
10    // pega o campo da imagem e armazena no array de informações
11    $arquivo = $_FILES['cpFoto'];
12    $info['media'] = "@$arquivo[tmp_name]";
13    // inicializa o recurso curl
14    $curl = curl_init();
15    // Define todos os parâmetros necessários ao objeto curl
16    curl_setopt($curl, CURLOPT_CONNECTTIMEOUT, 2);
17    curl_setopt($curl, CURLOPT_HEADER, false);
18    curl_setopt($curl, CURLOPT_RETURNTRANSFER, 1);
19    curl_setopt($curl, CURLOPT_BINARYTRANSFER, 1);
20    // Nesta etapa definimos a URL que será requisitada
21    // http://twitpic.com/api/upload - apenas posta a imagem e a mensagem
22    // http://twitpic.com/api/uploadAndPost - posta a imagem a mensagem e envia via Twitter
23     curl_setopt($curl, CURLOPT_URL, "http://twitpic.com/api/upload");
24     curl_setopt($curl, CURLOPT_POST, 3);
25    // envia os campos via post
26     curl_setopt($curl, CURLOPT_POSTFIELDS, $info);
27    // executa o objeto curl
28     $retorno = curl_exec($curl);
29     curl_close($curl);
30     // seta o header para um documento em XML
31    header('Content-type: text/xml');
32    // escreve o retorno obtido pelo curl ao enviar as informações para a URL informada
33    echo $retorno; ;
34 ?>
```

Na linha 03 criamos um array que irá armazenar todos os parâmetros que serão enviados via URL para o Twitpic. Entre as linhas 05 a 12 definimos os campos que serão enviados, como por exemplo, usuário e senha (nesta parte coloque o seu usuário e senha do Twitter, caso contrário não será feito login no sistema). Perceba a existência do campo media, que armazena a imagem e o campo message, que armazena a mensagem.

Na linha 14 começamos a configurar o recurso CURL. Este recurso nos permite que sejam definidas as propriedades informando como ele deve se comportar. Na linha 16 fizemos a definição do tempo que o CURL aguardará entre as requisições. As linhas 17, 18 e 19 definem as propriedades de header do recurso, ou seja, como CURL irá interpretar o retorno. Na linha 23 é que está um dos recursos mais importantes que é a definição de qual URL deverá ser chamada, perceba nos comentários das linhas 21 e 22 que podemos ter diferentes orlas para diferentes recursos, ou seja, dependendo de qual URL passarmos podemos ter a postagem da foto e da mensagem, mas não o envio via Twitter. Na linha 26 passamos como parâmetro o nosso array contendo as informações, é neste campo que realizaremos o post das mesmas.

Após todas as propriedades devidamente definidas, realizamos a execução do recurso, que pode ser visto na linha 28, e gravamos o retorno na variável $retorno, que armazenará o XML retornado pelo Twitpic. Na linha 31 definimos o header de nossa página como sendo do tipo XML, ou seja, o que está na página é um documento XML válido e, por fim, escrevemos o XML retornado através da variável retorno.

Nota do DevMan

Por padrão o XAMPP vem com a biblioteca CURL desabilitada. Para habilitar é bastante simples, vá até o diretório de instalação do XAMPP (em nosso caso c:\xampp) e entre no subdiretório PHP, dentro deste diretório edite o arquivo php.ini. Encontre a linha ";extension=php_curl.dll" e retire o caractere ";" antes da linha ficando apenas "extension=php_curl.dll". Feito isso, salve o arquivo, feche o bloco de notas e reinicie o servidor apache.

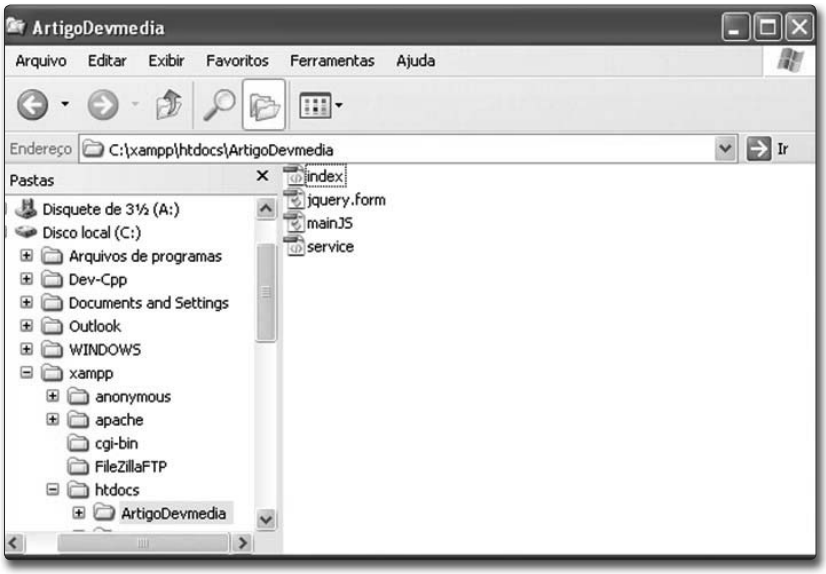


Figura 4. Estrutura de arquivos no diretório htdocs do XAMPP

Figura 5. Resultado final de nosso exemplo

Testando

Se você seguiu todas as instruções, devemos ter uma estrutura de arquivos semelhante à mostrada na **Figura 4**. O procedimento para testarmos é bastante simples, basta acessar o endereço http://localhost/ArtigoDevmedia, localizar uma imagem em seu computador e preencher os campos correspondentes, nossa foto será enviada ao Twitpic e a mensagem postada. Se você definiu a URL de envio e postagem no arquivo "service.php" uma mensagem será postada no seu Twitter e todos os seus seguidores poderão acessar. O resultado deverá ficar como na **Figura 5**.

Conclusão

As redes sociais são uma realidade para uma grande fatia dos internautas. Em consequência, recursos de integração com redes sociais são diferenciais valiosos em nossos sistemas, podendo ser determinantes na escolha de um projeto. Existe uma gama bastante grande de bibliotecas gratuitas para JavaScript, Prototype, por exemplo, e todas têm como finalidade facilitar a vida do desenvolvedor.

JavaScript independe da linguagem com a qual será utilizada. Lembre-se que você pode ser um desenvolvedor em ASP.NET e se valer dos recursos desta biblioteca, bem como Java ou ainda Rails, ou seja, não fique atrelado somente ao PHP, lembre-se que todas estas tecnologias geram HTML, e este pode ser manipulado via JavaScript e as bibliotecas já citadas.

Repare que em nosso exemplo utilizamos um tipo de codificação chamado JavaScript não obstrutivo, ou seja, não temos mistura de código e HTML, sendo que em nossa página inicial possuímos somente HTML puro, e todas as definições de funções e comportamentos estão separados da camada de visualização.

Twitter
http://www.twitter.com

Twitpic
http://www.twitpic.com

Download do plugin jquery.forms.js
http://glll.googlecode.com/files/jquery.form.js

Site oficial biblioteca jQuery
http://www.jquery.com

Manual PHP Curl
http://php.net/manual/en/book.curl.php

Site XAMPP
http://www.apachefriends.org/en/xampp.html
http://sourceforge.net/projects/xampp/files/

Blogs do autor
http://www.jaison.com.br
http://twitter.com/jaisonschmidt

Jaison Schmidt
dev@jaison.com.br
http://www.jaison.com.br
Autodidata em tecnologias, trabalha há mais de 12 anos com Web, desenvolvimento de sites, layouts e gerenciamento de conteúdo. Atualmente trabalha como Web Designer da Universidade de Passo Fundo no Rio Grande do Sul e é Acadêmico do Curso de Ciência da Computação no VIII nível. Autor de vídeo aulas para o portal DevMedia com mais de 80 vídeos publicadas.

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

*Sr. Cliente favor ler as informações com atenção, dúvidas entrar em contato com atendimento.

Relatório de Pré-Flight

OP: 85529 Cliente: Neofício Data: 26/08/10 Operador: Carlos

Informações da OP:
Título: **Clube Delphi**
Quantidade de Páginas: **CP+48**
Formato do Miolo: **205x275 mm**
Formato da Capa: **205x275 mm**
Formato da Lombada: *********
Tipo de Acabamento: **Canoa**

Informações do Arquivo:
Título: **Clube Delphi**
Quantidade de Páginas: **CP+48**
Formato do Miolo: **205x275 mm**
Formato da Capa: **205x275 mm**
Formato da Lombada: *********
Tipo de Acabamento: **Canoa**

Ocorrências do Arquivo

•Imagens com resolução abaixo de **200dpi**:

Pags. 6à10/12à18/20/21/23à26/29à47/50

•Imagens com resolução abaixo de **100dpi**:

Pags. 03/04/05 e Capa 02

•Texto Preto nas 4 Cores:

Pags. Capa 2 e 3

•Imagens em RGB nas páginas:

Pags. Capa 3

•Páginas sem sangria e/ou inferior a 3 mm: (Filete branco no acabamento)

Pag. 39

•Texto rente ao corte Inferior a 5 mm e/ou cortando:

•Colofon: **Não Tem**

• Código de Barra: **OK**

Observações:

OBS: As informações em **VERMELHO** necessitam de atenção, **se de acordo assinar.**

Data ____/____/____ Cliente: _____

Nome Legível